

**INTROSPECTIVE MULTISTRATEGY LEARNING:  
Constructing a Learning Strategy under  
Reasoning Failure**

A Thesis

Presented to

The Academic Faculty

by

**Michael Thomas Cox**

Tech. Rep. No. GIT-CC-96-06

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy in Computer Science

Georgia Institute of Technology

February, 1996

Copyright © 1996 Michael Thomas Cox

All Rights Reserved

Report Documentation Page			Form Approved OMB No. 0704-0188		
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
1. REPORT DATE <b>FEB 1996</b>		2. REPORT TYPE		3. DATES COVERED <b>00-00-1996 to 00-00-1996</b>	
4. TITLE AND SUBTITLE <b>Introspective Multistrategy Learning: Constructing a Learning Strategy under Reasoning Failure</b>			5a. CONTRACT NUMBER		
			5b. GRANT NUMBER		
			5c. PROGRAM ELEMENT NUMBER		
6. AUTHOR(S)			5d. PROJECT NUMBER		
			5e. TASK NUMBER		
			5f. WORK UNIT NUMBER		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) <b>Georgia Institute of Technology, Atlanta, GA, 30332</b>			8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSOR/MONITOR'S ACRONYM(S)		
			11. SPONSOR/MONITOR'S REPORT NUMBER(S)		
12. DISTRIBUTION/AVAILABILITY STATEMENT <b>Approved for public release; distribution unlimited</b>					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT <b>Same as Report (SAR)</b>	18. NUMBER OF PAGES <b>472</b>	19a. NAME OF RESPONSIBLE PERSON
a. REPORT <b>unclassified</b>	b. ABSTRACT <b>unclassified</b>	c. THIS PAGE <b>unclassified</b>			

**INTROSPECTIVE MULTISTRATEGY LEARNING:  
Constructing a Learning Strategy under  
Reasoning Failure**

Approved:

Ashwin Ram, Chair

---

Kurt Eiselt

---

Janet Kolodner

---

Nancy Nersessian

---

Margaret Recker

---

Tony Simon

---

Date Approved 02/23/96

## **DEDICATION**

To my father, Charles N. Cox, for teaching me honesty,  
to my mother, Betty M. Cox, for teaching me love,  
and to Scott Johnson, my first mentor, for teaching me to question.

## ACKNOWLEDGEMENTS

It tends to be a perfunctory courtesy for an author to acknowledge the contributions of colleagues in works as large as this one. But in my case, this debt is necessarily incurred because it is so difficult at times to distinguish the original work from that of so many talented people with whom I have had contact. For example, in countless discussions with my advisor, Ashwin Ram, it is often difficult to remember who had a particular idea first. Indeed, my interaction with Ashwin has been more like a grand synthesis of ideas than like a stereotyped conversation where participants exchange fully sovereign propositions. Social intercourse is by nature a mingling of ideas (a true “conception,” if I may use the metaphor), and so to identify the father of the idea is often impossible. Furthermore, I am almost convinced that there is nothing a writer can put to paper, the essence of which has not been said previously. One need only look hard enough and know the literature well enough to find the original references. Likewise, this thesis contains the germs of thought from many friends, colleagues, and distant siblings. Here I would like to acknowledge a few of those people who, although not directly quoted in the text, have nonetheless made a deep impression upon this work by their fresh infusion of ideas, support, criticism, and fellowship.

First and genuinely foremost, I must recognize Ashwin. For me, I could not have had a more perfect advisor. Ashwin has given me an especially long tether from which to work, while simultaneously challenging me to discipline myself. Although he has encouraged independence, he has also intervened at auspicious moments to point out corrections, clarifications, and insights when I needed them the most. This balance has been most rewarding, and I do not know where he has discovered the time to provide such leadership. He is one of the most prolific members of the faculty in the College of Computing, yet he always keeps an open door for discussing the issues as they arise. I also cannot fathom how he manages all of the students, projects, readings, publications, classes, and grant proposals, but his sheer volume of productivity has been an inspiration to us all, especially to me. He represents the true prototype from which the category *intellectual* is derived.

The truth is that Georgia Tech is a great place to become an intellectual and to do science (or engineering, depending on one’s persuasion). The environments, both within the AI Group and within the multi-college context of cognitive science at Tech, have made for a wonderful melange of experience. In particular, my committee has done much to support an interdisciplinary research emphasis within each area they represent. Janet Kolodner has inspired me from the beginning of my graduate career, and I owe much of my cross-disci-

plinary bias and interests to her. Without Janet, there never would have been a Cognitive Science Program at Georgia Tech, nor would I have become nearly as excited about the business of cognitive science and the issues it entails. I also thank Mimi Recker, especially for the Berkeley protocols and the long conversations we had on the subjects of cognitive science. Her mix of computer science, psychology and educational research has made me appreciate the complexity and the rewards of investigation that crosses the standard disciplinary boundaries. Tony Simon, Kurt Eiselt, and Nancy Nersessian have done likewise, each examining issues at the intersection between computing and psychology, computing and psycholinguistics, and computing and philosophy, respectively. All of these influences have given me a unique insight into some amazing and intriguing worlds that indeed surpass the best fiction.

The students involved in cognitive science have also had many influences. For instance, Chris Hale examined how humans troubleshoot device faults and make explanations, while Mike Byrne looked at everyday memory-errors. Discussions with both have helped make this thesis a more complete cognitive science work. In Industrial and Systems Engineering, the Center for Human-Machine Systems Research includes many people who participate in cognitive science as well. In particular, I wish to acknowledge the input of S. Narayanan (now at Wright State University) and Ellen Bass. But most of all, within the AI group and especially within Ashwin's IGOR research group, many friends have made Georgia Tech a more interesting and productive place to work. In particular, Mark Devaney has helped, not only with conceptualizations, but with the programming of the Tale-Spin and script application extensions of Meta-AQUA that enabled the empirical evaluation found in Chapter IX. Many others in the AI group have also contributed to the research, to reading chapters, and to the close friendships that exist between the students. Among these friends (but without constituting an exhaustive list) are Michael Pearce, Justin Peterson, Eleni Stroulia, Sam Bhatta, Kavi Mahesh, Kenny Moorman, Janis Roberts, Juan Carlos Santamaria, and Tucker Balch.

Many people in the College and elsewhere at Georgia Tech have made important impacts. Special thanks must be given to John Goda for originally encouraging me to come to Tech and Ed Rumiano for encouraging me to enter the doctoral program. Included with these two, I must also mention Gus Baird, Richard Billington, Stan Carpenter, Sue Farrell, Bud Foote, Peter Freeman, Chris Hertzog, Cindy Hmelo, Carolyn Russell, Tim Salthouse, and Andy Smith. Outside of Tech many people have also made contributions, and I wish to briefly list them: Larry Barsalou, Robert Cox, Eric Domeshek, John Dunlosky, Michael Freed, Tom Hinrichs, David Leake, Joel Martin, Rüdiger Oehlmann, Michael Pazzani, Foster Provost, Mike Redmond, and Philip Siegmann. And finally, I thank Allyana Ziolkow for karmic encouragement, a category that transcends both space and time.

But despite all of these significant contributions, this dissertation might never have come to fruition if it were not for the tireless effort and understanding of my wife Jennifer

Snow Wolff Cox. She has tolerated my impatience, soothed my irritation, and been there when I needed her the most. She has also found time to contribute directly to this thesis. It was Jennifer who suggested the quotation from Alexander Pope, who drew all of the robot illustrations, who proofed the entire document, and who provided a constant stream of suggestions that improved both the content and the presentation (while concurrently attending to her own work and ambitions, e.g., finishing her Masters). She has spent hours working on details such as the bibliography and index to make this thesis a complete work. Thank you for everything, Jennifer, especially for being yourself during my mad epic.

This research has been supported by the National Science Foundation under grant number IRI-9009710, by the Air Force Office of Scientific Research under grant number F49620-94-1-0092 and by the Georgia Institute of Technology. I also thank King Features Syndicate, New York, for special permission to reprint a strip of the “Walnut Cove” cartoon. Elizabeth Nolan was especially helpful and kind in facilitating the reprint request. This document was prepared almost entirely with FrameMaker (Version 4) under Unix on an IBM RS6000. Some figures were created using Draw on a NeXT Machine and Excel on a Macintosh. The graphs were created with gnuplot. I also used emacs and the fm2html filter to create a hypertext version of this document at the following universal resource locator (URL) on the World Wide Web:

`ftp://ftp.cc.gatech.edu/pub/ai/ram/git-cc-96-06.html`





## PREFACE

This thesis constructs a theory of introspective multistrategy learning. In large part, the work represents a machine learning theory in the area of multistrategy systems that investigates the role of the planning metaphor as a vehicle for integrating multiple learning algorithms. To another extent, the research is a cognitive science treatise on a theory of introspective learning that specifies a mechanistic account of reasoning about reasoning failure. The central idea is to represent explicitly the normal reasoning of an intelligent system in specific knowledge structures. When failure occurs, the learner can then examine the structures to explain what went wrong and hence to determine the proper learning methods. Thus, the overarching goal of the theory is to understand systems that turn inwards upon themselves in order to learn from their own mistakes.

I first became interested in reflective systems that process representations of themselves when working in the laboratory of Larry Barsalou. He and Chris Hale were building a theory of explanation in humans within the domain of troubleshooting small engine mechanics. I spent a year programming a system called MECH (Barsalou, Hale & Cox, 1989) that was designed to present domain knowledge about lawn mower engines, test their troubleshooting ability, and collect reaction times and other responses. In addition to its data collection mode, the program had a training mode with which information could be presented to the student before the test phase of a given experiment.

The MECH system had the ability to read data files that contained the entire domain theory: the engine system-subsystem decomposition, the test and repair screens, user selection functions, and associated textual data. With these files the experimenter could build domain-independent informational systems, not simply variations about engines. Included in the file formats was a means for specifying graphics screens that would be displayed at each subsystem along with the associated text screen for the user. Thus, the user could traverse the data base along graphical links to move from the fuel subsystem to the details of the carburetor component within that system, for example.

So, given this flexible capability at the end of the project, I conceived the idea to compile all of the comments for the program code and its subsystem modules. I designed a number of graphics screens that illustrated the hierarchical structure of the modules, functions, data structures and code. I then bound the appropriate program comments to the associated graphics screens using the generic file formats with which Chris had built the

engine explanation system. I could then turn MECH upon its own descriptions so that it would explain itself to Larry or Chris or any subsequent programmer they hired. The concept was simple and intriguing and stayed with me until this time.

This thesis poses the question “How can a reasoner create a learning strategy when it fails at its reasoning task?” The problem is called, simply enough, the learning-strategy construction problem. As a metaphor, consider a lawn mower. When the lawn mower breaks down, someone has to repair it so that it will work right in the future. Strategy construction is like choosing the right tools from the a tool box and planning how to use them to fix the broken lawn mower (see Figure 1). The person who does the repair is usually the owner, that is, if the task is not too hard. One of the things that the repairman has to worry about, however, is that the sequences of repair steps must be ordered properly so that they do not interfere with the overall goal of fixing the machine. For example, if the lawn mower runs out of gas because it has a small hole in the gas tank, the repairman must weld the damaged tank *before* adding gasoline to the tank. If the reverse order is chosen, then the gas will drain out of the tank and the entire machine might ignite (if not explode) when welding the fuel tank.

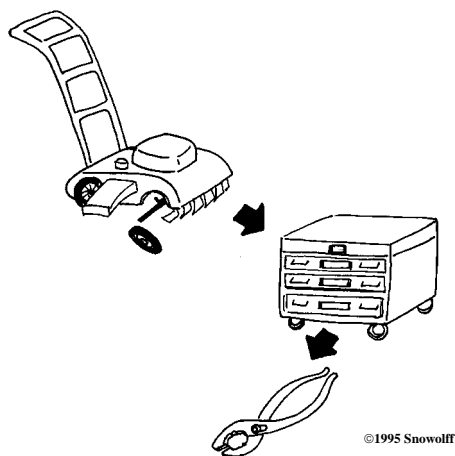


Figure 1. The strategy construction problem

Returning to the task of repairing the knowledge base of the reasoner, the problem is to choose and combine a few learning algorithms, given some suite of standard fixes, in order to fix the knowledge base of the system. Researchers usually perform this task. But in the research presented here, we want to automate the task of choosing the algorithms and let the machine solve the problem autonomously and dynamically. For the task of fixing the lawn mower, this is like putting a robot on top of the toolbox (see Figure 2).

When it comes to picking tools to fix the lawn mower, the task is usually straightfor-

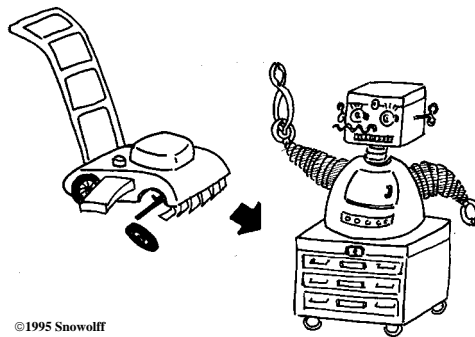


Figure 2. Automating the construction task

ward. The symptom of failure is often closely associated with the fix because the fault that caused the symptom is directly observable. For example, the lawn mower does not mow straight because the wheel has come loose. Because the cause of the error is readily apparent, the tool for the job is easy to determine. The task of assigning blame to a given failure is trivial because the cause is directly connected to the symptom.

But at other times, determining the cause of failure is much more problematic because the symptom of failure is more indirectly related to the fault. One cannot always tell what is wrong (and thus what needs to be repaired) by simply looking at the mower. If the lawn-mower makes a noise and the grass is not cut properly (see Figure 3), then the problem of assigning blame is that the fault must be inferred from the outcome of the performance task (i.e., grass cutting), rather than being directly observed. From a trace of the mower's path, the reasoner has to figure out what went wrong.

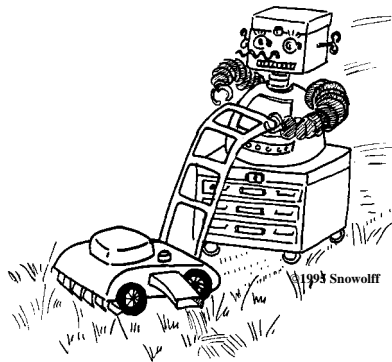


Figure 3. The blame-assignment problem

When trying to reason about what goes wrong in an intelligent system, the problem is compounded. A cognitive system is much more complicated than a simple lawn mower

and the chain of events between the initiation of a train of thought and its outcome is longer and more convoluted. With most intelligent systems no observables exist except the conclusions from long series of inferences. Therefore it is imperative that an explicit representation for the reasoning be present so that the system can “observe” it.

Moreover, even with mowing the lawn, the problem may be not so much with the device that cuts the lawn, but with the agent who cuts it (see Figure 4). That is, the reason the lawn is cut poorly may reside with the ability of the one doing the pushing. In our case, the fault may reside with the knowledge with which inferences are made rather than with the form of the logic used to make the inference.

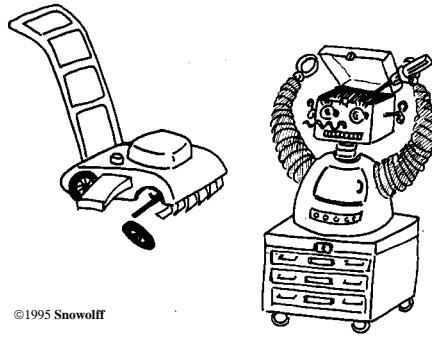



---

Figure 4. Cognitive causes of failure

Therefore, to repair the situation so that the lawn is cut better in the future, it may be necessary to fix the knowledge that “drives” the performance task (the robot), rather than to fix the performance system itself (the mower). Figure 5 illustrates this point. This entails understanding the causes of the failure, explaining what went wrong with the reasoning that caused the failure, and knowing enough about one’s own knowledge and the tools of knowledge repair to choose the right tools to fix the problem without letting these tools interfere with each other.

This thesis explores these issues from many perspectives. The intent is to look at the problem from both a technical and computational standpoint where we can analyze the representations and transformations useful in solving it mechanistically and to examine it from a synoptic and psychological standpoint to glean a bit of the human gestalt involved.



---

Figure 5. How to repair the failure?



## TABLE OF CONTENTS

DEDICATION . . . . .	iv
ACKNOWLEDGEMENTS . . . . .	v
PREFACE . . . . .	ix
TABLE OF CONTENTS . . . . .	xv
LIST OF TABLES . . . . .	xxi
LIST OF ILLUSTRATIONS . . . . .	xxiii
LIST OF ABBREVIATIONS . . . . .	xxvii
SUMMARY . . . . .	xxix

### *Part One PRELIMINARIES*

CHAPTER I INTRODUCTION . . . . .	3
1.1 Research Motivation . . . . .	4
1.2 The Problem . . . . .	7
1.3 The Solution . . . . .	9
1.4 Learning Goals and the Decomposition of the Problem . . . . .	12
1.4.1 The Learning Goal Metaphor . . . . .	13
1.4.2 The Decomposition of a Learning Goal . . . . .	15
1.4.2.1 What kinds of reasoning failure exist? (Q6) . . . . .	16
1.4.2.2 What can cause reasoning failure? (Q4) . . . . .	17
1.4.2.3 At what level of granularity should reasoning be represented? (Q7) . . . . .	18
1.4.2.4 How to represent mental states and reasoning mechanisms? (Q5) . . . . .	18
1.4.2.5 How to explain a reasoning failure? (Q3) . . . . .	19
1.4.2.6 How to decide what to learn? (Q2) . . . . .	20
1.4.2.7 How to choose or construct a learning strategy? (Q1) . . . . .	21
1.4.2.8 How can the research be evaluated? (Q0) . . . . .	21
1.5 Overview of the Dissertation . . . . .	23
CHAPTER II CONTENT THEORIES AND PROCESS THEORIES . . . . .	25
2.1 The Drug-Bust Examples . . . . .	26
2.1.1 A Common Contradiction . . . . .	26
2.1.2 A Baffling Situation . . . . .	28
2.2 Knowledge and Process . . . . .	30
2.3 The Domain of Story-Understanding Failures . . . . .	32
2.4 Prolog to Parts Two and Three: The content and the process . . . . .	37

*Part Two A CONTENT THEORY OF MENTAL REPRESENTATION*

CHAPTER III SYMPTOMS AND CAUSES OF FAILURE: The Content . . . .	41
3.1 A General Model of Expectation-Driven Reasoning . . . . .	43
3.2 Types of Reasoning Failure . . . . .	45
3.2.1 Four Basic Cases . . . . .	46
3.2.1.1 Contradiction . . . . .	46
3.2.1.2 Impasse . . . . .	47
3.2.1.3 False expectation . . . . .	47
3.2.1.4 The degenerate case . . . . .	48
3.2.2 Extending the Analysis . . . . .	48
3.2.2.1 Surprise . . . . .	49
3.2.2.2 Unexpected success . . . . .	49
3.3 Causal Factors in Reasoning Failure . . . . .	50
3.3.1 Selection Factors . . . . .	51
3.3.2 A Taxonomy of Reasoning Failure Causes . . . . .	52
3.3.3 Knowledge States . . . . .	55
3.3.3.1 Domain knowledge . . . . .	55
3.3.3.2 Knowledge selection . . . . .	56
3.3.4 Goal States . . . . .	59
3.3.4.1 Goal generation . . . . .	59
3.3.4.2 Goal selection . . . . .	60
3.3.5 Processes . . . . .	61
3.3.5.1 Processing strategy . . . . .	61
3.3.5.2 Strategy selection . . . . .	62
3.3.6 Environment . . . . .	62
3.3.6.1 Input . . . . .	63
3.3.6.2 Input selection . . . . .	63
3.3.7 Causal Invariants . . . . .	64
3.4 Summary and Discussion . . . . .	65
CHAPTER IV MENTAL STATES AND MECHANISMS: The Representation . .	67
4.1 Epistemology and Ontology . . . . .	68
4.2 Granularity of Representation . . . . .	69
4.3 Representing Forgetting: An example . . . . .	72
4.3.1 Logic . . . . .	73
4.3.2 Conceptual Dependency . . . . .	74
4.3.3 Explanation Pattern Theory . . . . .	75
4.4 Meta-Explanation Patterns . . . . .	79
4.4.1 Trace Meta-XPs . . . . .	81
4.4.2 Introspective Meta-XPs . . . . .	84
4.5 Vocabulary . . . . .	90
4.6 Representing Reasoning Success . . . . .	91



4.7 Representing Reasoning Failure to Support Learning . . . . .	.93
4.7.1 Contradiction . . . . .	.95
4.7.2 Impasse . . . . .	.95
4.7.3 False Expectation . . . . .	.96
4.7.4 Surprise . . . . .	.98
4.7.5 Unexpected Success . . . . .	.98
4.8 Summary and Discussion . . . . .	.98

### *Part Three A PROCESS THEORY OF LEARNING AND INTROSPECTION*

## CHAPTER V A PROCESS THEORY OF UNDERSTANDING AND LEARNING 105

5.1 Theoretical Assumptions . . . . .	.105
5.2 Multistrategy Reasoning . . . . .	.107
5.3 Process Model of Understanding . . . . .	.109
5.3.1 Three Sub-processes of Understanding . . . . .	.109
5.3.2 Understanding Elvis' Behavior . . . . .	.111
5.3.3 Question-Driven Understanding . . . . .	.112
5.4 Process Model of Learning . . . . .	.115
5.4.1 Multistrategy Learning . . . . .	.117
5.4.2 Process Divisions within the Model of Learning . . . . .	.120
5.4.3 Generating Changes to the BK . . . . .	.122
5.4.3.1 Functional justification for introspection as a component of learning	123
5.4.3.2 Overview of the IML algorithm . . . . .	.125
Blame assignment . . . . .	.125
Deciding what to learn . . . . .	.127
Learning-strategy construction . . . . .	.127
5.5 Comparison of Learning and Understanding . . . . .	.128
5.6 Summary . . . . .	.130

## CHAPTER VI CASE-BASED INTROSPECTION . . . . . 133

6.1 Case-Based Reasoning and Introspection . . . . .	.134
6.2 Blame Assignment: Explaining reasoning failure . . . . .	.136
6.2.1 The Explanation Pattern Application Algorithm . . . . .	.137
6.2.2 Returning to the Drug Bust Example . . . . .	.140
6.3 Deciding What to Learn: Spawning learning goals . . . . .	.143
6.3.1 Explicit Learning Goals . . . . .	.146
6.3.2 Learning Goals in the Drug-Bust Example . . . . .	.149
6.4 Summary and Discussion . . . . .	.151

## CHAPTER VII LEARNING AS A NON-LINEAR PLANNING TASK . . . . . 153

7.1 Blame Assignment and Repair: Tight versus loose coupling . . . . .	154
7.1.1 Direct Mappings: Tight coupling . . . . .	155
7.1.2 Indirect Mappings: Loose coupling . . . . .	157
7.2 Learning-Strategy Construction: Learning as a planning task . . . . .	158
7.2.1 Managing Goal Interactions with Nonlinear Planning . . . . .	159
7.2.2 Planning for Blocks World Goals . . . . .	160
7.2.3 Planning for a Knowledge Differentiation Goal . . . . .	162
7.2.4 Planning for a Knowledge Reconciliation Goal . . . . .	165
7.3 Strategy Execution: Performing the learning and the aftermath . . . . .	168
7.4 The Planning Metaphor in Learning . . . . .	168
7.4.1 Generality of the Metaphor . . . . .	170
7.4.2 Advantages of the Planning Metaphor . . . . .	171
7.5 Summary and Discussion . . . . .	172

## *Part Four IMPLEMENTATION AND CONCLUSION*

## CHAPTER VIII META-AQUA . . . . . 177

8.1 Meta-AQUA System Architecture . . . . .	178
8.2 The Performance Subsystem . . . . .	180
8.3 Input Subsystem . . . . .	184
8.3.1 The Tale-Spin Story-Generator . . . . .	184
8.3.2 The Elvis World . . . . .	185
8.3.3 Interface to the Performance System . . . . .	188
8.4 Memory . . . . .	188
8.4.1 Types and Tokens . . . . .	190
8.4.2 Indexing . . . . .	192
8.4.3 Reminders in Opportunistic Memory . . . . .	194
8.5 Learning Subsystem . . . . .	196
8.5.1 Divisions of the Learner . . . . .	197
8.5.2 The Implemented Space of Explanation Failures . . . . .	198
8.5.3 Learning about higher-order knowledge . . . . .	201
8.5.4 Forgetting a Learned Explanation . . . . .	203
8.6 Summary . . . . .	209

CHAPTER IX EVALUATION . . . . .	211
9.1 The Hypotheses . . . . .	213
9.1.1 Hypothesis Number 1: Introspection facilitates learning . . . . .	214
9.1.1.1 Fully introspective and semi-introspective learning . . . . .	215
9.1.1.2 The null hypothesis . . . . .	216
9.1.2 Hypothesis Number 2: IML theory is a sufficient model of introspection	217
9.2 Computational Evaluation . . . . .	219
9.2.1 The Experimental Conditions (Independent Variable) . . . . .	219
9.2.2 The Evaluation Criteria (Dependent Variables) . . . . .	220
9.2.3 The Empirical Data . . . . .	221
9.2.3.1 Run Number Four . . . . .	222
9.2.3.2 Quantitative Results . . . . .	224
9.3 Psychological Plausibility . . . . .	229
9.3.1 Learning to program in LISP . . . . .	230
9.3.2 Meta-AQUA Simulation . . . . .	231
9.3.3 Learning to Troubleshoot Circuit Boards . . . . .	236
9.4 Summary and Discussion . . . . .	241
CHAPTER X FUTURE RESEARCH . . . . .	245
10.1 IML Theory and Implementation: Open issues . . . . .	246
10.1.1 The Scope of Blame-Assignment. . . . .	246
10.1.2 Additional Goal Interactions. . . . .	247
10.1.3 Enlarging the Toolbox. . . . .	248
10.1.4 Parallelism in Learning Plans . . . . .	249
10.1.5 Learning New IMXPs. . . . .	249
10.1.6 Auto-Validation of Learning. . . . .	250
10.1.7 Failure Types as a Cognitive Category . . . . .	251
10.1.8 Extending the Computational Evaluation . . . . .	252
10.1.8.1 Establishing the hypothesis that introspection facilitates learning . . . . .	252
10.1.8.2 Determining the conditions under which the hypothesis holds . . . . .	252
10.1.9 Extending the Model of Human Learning . . . . .	253
10.1.10 Extending the Representational Vocabulary . . . . .	253
10.2 Learning Bias and Category Membership: An extension . . . . .	255
10.3 Understanding student explanation failures: An application . . . . .	257
10.4 Summary . . . . .	261
CHAPTER XI RELATED RESEARCH . . . . .	263
11.1 Artificial Intelligence, Metareasoning, and Introspective Learning . . . . .	264
11.1.1 Logic and Belief Introspection . . . . .	265
11.1.2 Knowledge-Based Systems, Quasi-Introspection, and Related Theories	267
11.1.3 Relation of AI Research to IML Theory . . . . .	269
11.2 Psychology, Metacognition, and Human Learning . . . . .	272

11.2.1 Cognition and Metacognition . . . . .	272
11.2.2 Problem Solving and Metacognition . . . . .	273
11.2.3 Metamemory . . . . .	275
11.2.4 Relation of Psychological Research to IML Theory . . . . .	277
11.3 Summary and Discussion . . . . .	280
CHAPTER XII CONCLUSIONS . . . . .	283
12.1 Major Points of the Dissertation . . . . .	284
12.2 Part One: Motivations and Defining the Problem . . . . .	285
12.3 Part Two: Content Theory of Introspective Multistrategy Learning . . . . .	287
12.4 Part Three: Process Theory of Introspective Multistrategy Learning . . . . .	288
12.5 Part Four: Evaluation and Implementation of the Theory . . . . .	289
12.6 Contributions . . . . .	290
CHAPTER XIII EPILOGUE . . . . .	293
13.1 The Processes:	
Integration of problem solving, understanding, and learning . . . . .	294
13.2 The Knowledge:	
Integration of world knowledge, metaknowledge and self-knowledge . . . . .	299
13.3 The "Turing Test" . . . . .	302
 <i>APPENDICES, REFERENCES AND INDEXES</i> 	
APPENDIX A The Degrees of Freedom in Learning . . . . .	311
APPENDIX B Meta-AQUA Output in Story Understanding Mode . . . . .	315
APPENDIX C Story Listings for Run Number Four . . . . .	345
APPENDIX D Meta-AQUA Output in LISP Programming Mode . . . . .	359
REFERENCES . . . . .	375
NAME INDEX . . . . .	411
PROGRAM INDEX . . . . .	419
SUBJECT INDEX . . . . .	421
VITA . . . . .	441

## LIST OF TABLES

Table 1: Basic taxonomy of causes of reasoning failure . . . . .	17
Table 2: Logical truth table for reasoning model . . . . .	46
Table 3: Expanded table for reasoning model . . . . .	48
Table 4: Final table for reasoning model . . . . .	50
Table 5: Detailed taxonomy of causes of reasoning failure . . . . .	53
Table 6: Truth values of graph nodes for forgetting . . . . .	78
Table 7: Structural differences between remembering events . . . . .	94
Table 8: Matrix associative-solution to strategy construction . . . . .	156
Table 9: Results from run number four . . . . .	223
Table 10: Main character distribution (run four). . . . .	224
Table 11: Problem distribution (run four) . . . . .	224
Table 12: Summary of results from run four. . . . .	225
Table 13: Summary of cumulative results . . . . .	229
Table 14: Extending IML theory . . . . .	246
Table 15: Is the plane neutral? Possible causes . . . . .	313



## LIST OF ILLUSTRATIONS

Fig. 1. The strategy construction problem .....	x
Fig. 2. Automating the construction task .....	xi
Fig. 3. The blame-assignment problem .....	xi
Fig. 4. Cognitive causes of failure .....	xii
Fig. 5. How to repair the failure? .....	xiii
Fig. 6. Andrew's failure .....	6
Fig. 7. Decomposition of the learning problem .....	9
Fig. 8. Basic Meta-AQUA system architecture .....	11
Fig. 9. Primary research subgoal tree and contributions .....	14
Fig. 10. Forgetting to fill the tank with gas .....	19
Fig. 11. Hand-coded story HC1 (from Cox & Ram, 1991) .....	27
Fig. 12. Hand-coded story HC2 (from Cox, 1994b) .....	28
Fig. 13. Multilevel representations and processes .....	34
Fig. 14. Question-driven understanding .....	36
Fig. 15. A reasoner's input .....	43
Fig. 16. The basic comparison model .....	44
Fig. 17. The extended comparison model .....	49
Fig. 18. Example generalization hierarchy for color .....	57
Fig. 19. Preliminary partial ontology of mental terms .....	70
Fig. 20. CD representation of abstract transfer (ATRANS) .....	71
Fig. 21. CD representation of forgetting .....	75
Fig. 22. XP representation of XP-GOAL-OF-OUTCOME->ACTOR .....	76
Fig. 23. Meta-XP representation of forgetting .....	77
Fig. 24. XP as a directed graph .....	80
Fig. 25. Relations as first-class objects .....	81

Fig. 26. Volitional XP for why agent performs suicide bombing .....	82
Fig. 27. Graph structure for Decide-Compute-Node .....	85
Fig. 28. Frame definition for Decide-Compute-Node .....	86
Fig. 29. Representation for forgotten detection explanation .....	88
Fig. 30. IMXP frame definition for forgetting .....	89
Fig. 31. Meta-XP representation of successful prediction .....	91
Fig. 32. Successful prediction frame definition .....	92
Fig. 33. Meta-XP representation of several remembering events .....	94
Fig. 34. Meta-XP representation of contradiction .....	95
Fig. 35. Meta-XP representation of impasse .....	96
Fig. 36. Meta-XP representation of false expectation .....	97
Fig. 37. Meta-XP representation of surprise .....	99
Fig. 38. Meta-XP representation of unexpected success .....	100
Fig. 39. CD representation of expectation .....	101
Fig. 40. Assumptions .....	106
Fig. 41. Basic reasoning model .....	109
Fig. 42. Sub-processes of understanding .....	110
Fig. 43. Question-driven understanding .....	113
Fig. 44. Traditional model of learning .....	116
Fig. 45. Model of introspective multistrategy learning .....	119
Fig. 46. Sub-processes of learning .....	120
Fig. 47. Failure detection algorithm .....	121
Fig. 48. IML learning algorithm .....	126
Fig. 49. Parallels between learning and understanding .....	129
Fig. 50. Reflective blame assignment .....	138
Fig. 51. Representation of the question “Is Z believable?” .....	139
Fig. 52. The drug-bust story (HC1) .....	140
Fig. 53. Meta-AQUA output during hypothesis generation of HC1 .....	141
Fig. 54. Instantiated IMXP for mis-explanation .....	142



Fig. 55. IMXP abstraction of causal factors involved in story HC1 .....	144
Fig. 56. Learning output and frame representation of Meta-XP used in example story .....	145
Fig. 57. Broad goal categories .....	146
Fig. 58. Frame definition for goals .....	147
Fig. 59. A taxonomy of learning goals .....	148
Fig. 60. Two learning goals spawned in story HC1 .....	150
Fig. 61. The Sussman anomaly .....	160
Fig. 62. Blocks world operators .....	161
Fig. 63. Abstracted IMXP with learning goals .....	163
Fig. 64. Schema definitions to index an XP .....	164
Fig. 65. Mutual-indexing schemas .....	166
Fig. 66. Abstraction schema .....	167
Fig. 67. Story HC3: Another hidden stash .....	168
Fig. 68. Nonlin output and the final learning-plan .....	169
Fig. 69. Story HC1': The handball game .....	170
Fig. 70. Detailed Meta-AQUA system architecture .....	179
Fig. 71. Meta-AQUA file system definition .....	181
Fig. 72. Meta-AQUA flow of control .....	183
Fig. 73. Tale-Spin story TS1 .....	187
Fig. 74. Sentences from story TS1 corresponding to HC1 .....	187
Fig. 75. Representational flow between generator and understander .....	189
Fig. 76. Generalized frame token structure .....	191
Fig. 77. Indexing for items about why dogs bark .....	193
Fig. 78. Tale-Spin story TS2 .....	195
Fig. 79. Implemented space of reasoning faults .....	199
Fig. 80. Tale-Spin story TS3 .....	202
Fig. 81. Tale-Spin story TS4 .....	204
Fig. 82. Meta-AQUA output during hypothesis generation phase of TS4 .....	205
Fig. 83. Instantiated forgotten detection explanation .....	206

Fig. 84. Subsequent test story (HC3) .....	207
Fig. 85. Learning phase during TS4 .....	208
Fig. 86. Learning goal ablation .....	215
Fig. 87. Alternate ablations .....	217
Fig. 88. Run 4, cumulative question points as a function of the number of problems ..	226
Fig. 89. Run 4, question points histogram .....	226
Fig. 90. Run 4, cumulative correct answers (accuracy) as a function of the number of problems .....	228
Fig. 91. Run 4, correct answers (accuracy) histogram .....	228
Fig. 92. Strategy protocol categories .....	232
Fig. 93. Protocol fragment of subject AK88 .....	234
Fig. 94. Representation for interaction of learning strategies of protocol fragment from subject AK88. ....	235
Fig. 95. Add1nums function definitions .....	236
Fig. 96. Frame definitions to represent newly learned programming knowledge .....	237
Fig. 97. Cumulative diagnostic accuracy .....	240
Fig. 98. Speed ratio of learning conditions to hand-coded condition .....	240
Fig. 99. The trick story .....	257
Fig. 100. Research goals and results (contributions) .....	286
Fig. 101. Relationship between understanding and planning .....	297
Fig. 102. A taxonomy of knowledge .....	300

## LIST OF ABBREVIATIONS

AAAI	American Association for Artificial Intelligence
ACT	Adaptive Character of Thought
AI	Artificial Intelligence
BK	Background Knowledge
CBR	Case-Based Reasoning
CD	Conceptual Dependency
D-C-Node	Decide-Compute-Node
EBG	Explanation-Based Generalization
FK	Foreground Knowledge
FOK	Feelings-Of-Knowing
IMXP	Introspective Meta-eXplanation Pattern
IML	Introspective Multistrategy Learning
JOL	Judgements of Learning
LISP	LISt Processing language
MBUILD	Mental BUILD of conceptualization
Meta-XP	Meta-eXplanation Pattern
Meta-TS	Meta Trouble Shooter
MOP	Memory Organization Packet
MSL-91	MultiStrategy Learning workshop 1991
MTRANS	Mental TRANSfer of information
PDP	Parallel Distributed Processing
PFXP	Plan Failure eXplanation Pattern
SBF	Structure-Behavior-Function
SBL	Similarity-Based Learning

TMXP	Trace Meta-eXplanation Pattern
URL	Universal Resource Locator
XP	eXplanation Pattern

## SUMMARY

The thesis put forth by this dissertation is that introspective analyses facilitate the construction of learning strategies. Furthermore, learning is much like nonlinear planning and problem solving. Like problem solving, it can be specified by a set of explicit learning goals (i.e., desired changes to the reasoner's knowledge); these goals can be achieved by constructing a plan from a set of operators (the learning algorithms) that execute in a knowledge space. However, in order to specify learning goals and to avoid negative interactions between operators, a reasoner requires a model of its reasoning processes and knowledge. With such a model, the reasoner can declaratively represent the events and causal relations of its mental world in the same manner that it represents events and relations in the physical world. This representation enables introspective self-examination, which contributes to learning by providing a basis for identifying what needs to be learned when reasoning fails. A multistrategy system possessing several learning algorithms can decide what to learn, and which algorithm(s) to apply, by analyzing the model of its reasoning. This introspective analysis therefore allows the learner to understand its reasoning failures, to determine the causes of the failures, to identify needed knowledge repairs to avoid such failures in the future, and to build a learning strategy (plan). Thus, the research goal is to develop both a content theory and a process theory of introspective multistrategy learning and to establish the conditions under which such an approach is fruitful.

Empirical experiments provide results that support the claims herein. The theory was implemented in a computational model called Meta-AQUA that attempts to understand simple stories. The system uses case-based reasoning to explain reasoning failures and to generate sets of learning goals, and it uses a standard non-linear planner to achieve these goals. Evaluating Meta-AQUA with and without learning goals generated results indicating that computational introspection facilitates the learning process. In particular, the results lead to the conclusion that the stage that posts learning goals is a necessary stage if negative interactions between learning methods are to be avoided and if learning is to remain effective.



*Part One*

***PRELIMINARIES***





## CHAPTER I

### INTRODUCTION

*WURFSCHETBE, mit  
Vorgesichten besternt,  
wirf dich  
aus dir hinaus.*

—Paul Celan (1970), p. 41.

*DISCUS,  
starred with premonitions,  
throw yourself  
out of yourself.*

—translation (1986), p. 39.

This research investigates goal-driven learning by specifying a computational model of *introspective multistrategy learning (IML)*. The theory concerns introspection because learning in the model depends on the ability of the learner to reason about internal reasoning processes and mental states. To do this, a system must represent its knowledge about its own reasoning explicitly and declaratively in a tangible knowledge structure so that it can examine and manipulate it. It is a multistrategy theory because it is intended to integrate a wide variety of learning methods in a uniform manner. The specific focus of the research is on the *learning-strategy construction problem*. That is, given some computational performance task specified by the system's goals, context and some input, if a failure occurs during the task, the problem is to construct a learning strategy with which to repair the faulty components of the system. The solution to this problem is a hybrid model having two phases. In the event of a reasoning failure, the first phase uses case-based methods to retrieve declarative meta-explanation structures that support self-reflective blame assignment of the reasoning failure and that assist in the generation of a set of learning goals. Learning goals are necessary to mediate between the explanation of failure and the learning needed to avoid the failure in the future. Given such learning goals, the second phase of the model uses a non-linear planner to construct a partially ordered sequence of calls to specific learning algorithms to achieve the goals. The model implementation, called Meta-AQUA, illustrates our solution to the problem of selecting a learning algorithm in machine learning contexts. It is also used to simulate the learning performed by humans during complex reasoning tasks.

## 1.1 Research Motivation

An intelligent agent learns from its mistakes; a fool is doomed to repeat them. Thus, if one wants to understand intelligence, it is important to understand learning. Moreover, if one wishes to build either intelligent devices or models of intelligent agents, it is also important to give these systems the ability to learn from their experience, especially failed experiences, so that they can improve their performance over time and avoid repetition of failures. A system designer cannot hope to incorporate *ab initio* all the knowledge necessary for a nontrivial intelligent system, so learning provides a way to acquire or extend knowledge incrementally over time. Even supposing all knowledge could be incorporated into a system by some knowledge engineer, engineers occasionally make mistakes. Thus, learning will be required in order to detect and remove inconsistencies in the knowledge. Furthermore, even if a system has an exhaustively complete and consistent knowledge base, the world is dynamically changing. The system would therefore have to adjust its knowledge, otherwise it would soon become obsolete. But, beyond these engineering arguments, we are interested in understanding and building models of learning in order to discover a little bit about ourselves: about how we cope with mountains of information; about how we detect and retract mistaken assumptions and incorrect beliefs; and about how we adapt to the constantly changing world that surrounds us. Given this complex state of affairs, how can one best view the learning needed to institute these kinds of changes?

Consider, for example, a student trying to learn to program in the language LISP. Typically, one of the most difficult lessons to master is the concept of recursion. Despite having mastered the separate features of a programming language, when attempting to program a recursive function that incorporates known operations, students often fail. When a programming bug occurs, the student must both learn what went wrong with the specific recursive problem (i.e., recover from the current bug) and generalize and refine the concept of recursion (i.e., repair the student's knowledge so that bugs will not repeat in similar recursive problems). The student has a number of choices to perform these learning tasks. The student may return to a previous example, may re-read the textual instructions, may reflect over the problem solving that preceded the failure, or simply continue to the next problem with the hope that further problem solving will illustrate the proper method of solving such examples (Pirolli & Recker, 1994). Which strategy to use is a crucial decision that bears on the effectiveness of learning and thus the subsequent performance of the programmer.

To endow a machine with a similar strategic ability to learn, one might allow it to select a method from some library of learning algorithms when it makes reasoning errors during its performance task. For example, if a machine is designed to read simple stories, it might make mistakes when trying to understand the sentences in a story or when attempting to predict the actions or motivations of characters in a story. If it has read numerous stories about terrorism, it may use past cases of terrorist smuggling to understand an analogous story about drug smuggling. Yet its knowledge about events in the new story may

be incomplete or incorrect. When it reads about dogs that bark at suspicious luggage, it should be able to predict that the dog is barking in order to signal the presence of contraband. But if it does not have such knowledge, then it needs to be able to acquire experience about these events by creating a knowledge acquisition strategy consisting of various calls to routines in its library. Even if it has relevant cases, however, it may not actually be able to retrieve an appropriate case in a given situation while reading new stories. When it discovers this lack, it must be able to adjust the organization of its knowledge by constructing a memory reindexing strategy. IML theory directly addresses this problem of deliberately constructing a strategy with which to learn given some failure in its performance task.

When confronted by failure, a strategic learner must know what caused the failure and explain what went wrong in order to know what needs to be learned. Yet, the number of events that can mishap is immense. Therefore, an agent must not only decide on a strategy to learn, but first, the agent must sift through a large number of explanations that may account for any given failure. For instance, consider the Walnut Cove cartoon in Figure 6.

In this cartoon, Andrew's brother sneaks into his bedroom to wake him. He starts to scream that since it is already 8 o'clock in the morning Andrew will be late for school if he does not get ready quickly. As they run down the stairs, Andrew is asked whether or not he is forgetting anything important, to which he replies that he believes so. With a bit more thought, Andrew decides that he has definitely forgotten something, but he still cannot recall what it might be. Finally, as Andrew tries to clear his head at the bus stop, we discover Andrew's problem. It is Sunday, so the bus is not in service.

Waiting outside, Andrew will probably be wondering why the bus is late. To explain the bus's failure to arrive on time, Andrew might reason about the physical operation of the bus from some naïve model of automobiles and engines. He might conclude that the driver ran out of gas either because of a hole in the gas tank or because of some other mechanical failure. As this cartoon illustrates, however, the real problem is with Andrew's memory system, rather than with the bus's engine system. But, his memory is not the only causal factor that bears on his mistake. A major contribution to his memory problem was caused by his brother. By providing misinformation ("you're going to be late for school"), his brother gave him the wrong context from which to reason initially. Finally, Andrew seriously needs to consider his goals. The main problem is that Andrew is pursuing an inappropriate goal given his circumstances.

We see that the causes of failure may come from many sources; not only will failure occur from physical sources in the world, but Andrew must be able to look within himself to consider additional factors such as his own knowledge, memory, inferences, input, context and goals. Andrew does start to consider this; it is significant that Andrew knew that he was forgetting something, even though he could not remember what it was. Only after he knows what went wrong can Andrew learn not to repeat this mistake, perhaps by not

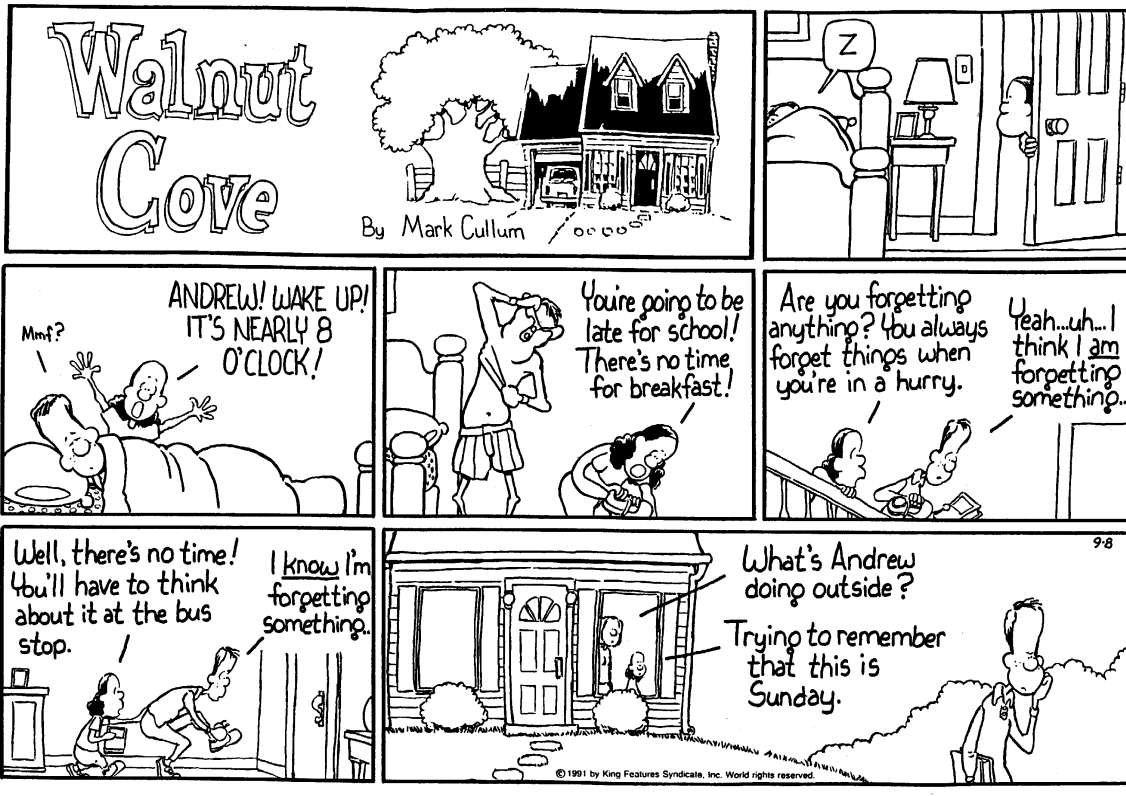


Figure 6. Andrew's failure  
(Reprinted with special permission of King Features Syndicate)

trusting his brother when he first wakes up in the morning, or perhaps by evaluating his goals more carefully.

These types of learning problems are ubiquitous. They confront not only the reasoner who tries to solve problems in some world, such as Andrew trying to decide his morning activities, but also agents trying to understand other agents like themselves. For example, in order for a reader of Walnut Cove to fully comprehend the cartoon's story, the agent must be able to understand Andrew's failure in terms of the mental events within Andrew, not simply the physical events drawn within the strip. Moreover, not only must readers be attuned to factors that affect failures performed by the characters of a story, but in order to improve their reading skills, readers must also be able to reason about their own comprehension failures when they incorrectly predict the twists and turns of a story. In both problem-solving tasks like daily planning and comprehension tasks like story understanding, a reasoner must be able to explain reasoning failure in order to construct some coherent strategy with which to learn.

## 1.2 The Problem

Simply stated, and in the narrowest sense, the central problem addressed by this research is the learning-strategy construction problem (Cox & Ram, 1991), particularly, in failure-driven learning. That is, given some goal-specific performance task (e.g., story understanding or problem solving), a context and some input, if a failure occurs during the task, the computational learning problem is to choose or construct a learning strategy with which to repair the background knowledge of the system.<sup>1</sup> The knowledge is considered repaired if, given a similar future situation, the failure will not recur. Yet, as seen in the previous section, the problem is not a simple one. In order to fix the knowledge effectively, the learner must first understand both the knowledge it is fixing and the error that gave rise to the need to learn. Since failure often is caused by faulty reasoning, the learner must be able to represent, examine, and reason about its own reasoning. In the broadest sense, then, this thesis attempts to carve out a theory of introspection and self-understanding.

Yet, it is not immediately apparent why introspection is necessary, or even desirable, in many cases. Introspection has the distinct disadvantage of considerable computational

---

1. The construction task may be as simple as choosing an algorithm from a list or may involve constructing a complex plan of sequenced learning steps that constitute the strategy. Chapter VII will provide details. The background knowledge of the system is not just a repository for the domain theory. It contains declarative representations for all long term knowledge, such as conceptual categories, episodes and cases, control knowledge (heuristics), beliefs, and knowledge about its own knowledge and reasoning processes (metaknowledge).

overhead. Furthermore, it is a well-founded fact that the veracity of human introspection is very limited.<sup>2</sup> In general, however, adding introspection to a machine allows it to have an idea of what it is doing and why. A machine applying deductive theorem-proving certainly does not understand mathematics in the same manner that a mathematician does. Because no model of the problem-solving process exists in an automatic theorem prover, the machine does not understand theorem proving even though it can perform it. We do not claim that introspection is a computational panacea; rather, this research investigates the role of introspection when constructing a learning strategy and the contingencies under which it is beneficial.

The problem of strategy construction is quite challenging because to construct a strategy, a system needs to know specifically what is supposed to be learned; to decide what needs to be learned, it must know the cause of failure; and to determine the full cause of the failure, it must be able to reflect upon its own reasoning. Thus, three major problems exist when facing a reasoning failure (Ram & Cox, 1994): blame assignment, deciding what to learn, and strategy construction. Figure 7 illustrates the relationships between these problems in graphical form. As will be explained shortly, the first two are case-based reasoning problems, whereas the third is a non-linear planning problem.

1. ***Blame Assignment*** – explain the misunderstanding by mapping from the symptom of the failure to the cause of the failure;
2. ***Decide What to Learn*** – form a set of explicit goals to change the knowledge so that such a misunderstanding is not repeated in similar situations;
3. ***Strategy Construction*** – construct a learning plan by which to achieve these goals.

From a machine learning perspective, serious obstacles exist when confronting the strategy construction problem. Many learning algorithms have been developed in the past thirty years of machine learning research, so there may be many options and algorithms from which to choose. Under the rubric of multistrategy learning research (e.g., Michalski & Tecuci, 1994), recent trends aim at incorporating the multiplicity of algorithms into a cohesive whole in which multiple strategies can be combined to tackle complex learning

---

2. In fact, evidence exists that introspection can actually degrade performance in skilled tasks that require judgements (Wilson & Schooler, 1991) and that in general, people are overly-confident in cognitive tasks such as question answering (Fischhoff, Slovic, & Lichtenstein, 1977). Also, see the criticism of early psychological studies that used trained introspection as a methodological tool (e.g., Boring, 1953).

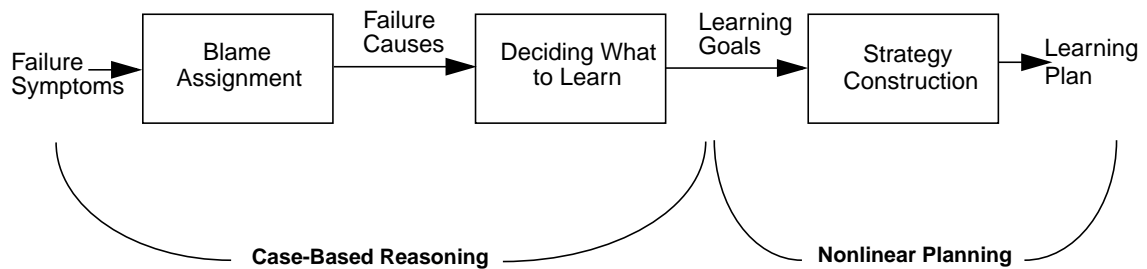


Figure 7. Decomposition of the learning problem

problems. In the current state of the art in machine learning research, however, humans analyze characteristic performance parameters (i.e., the particular input context and distribution of learning problems) and then decide what algorithms or combination of algorithms are best suited to the data. A major goal of this research is to begin to automate this process. The wish is to give machines a library of learning methods and have them independently decide which should be used to fix the problems that inevitably occur.

As indicated by the problems facing novice LISP programmers discussed in Section 1.1, there is also a cognitive science interpretation to this problem. Rather than merely formulating a method by which to engineer the machine learning problems in strategy construction, we are interested in developing a model that applies to humans engaged in deliberation over learning choices. Considering known human limitations from the psychological literature imposes realistic constraints on the model. The resulting cognitive model is not always easy to separate from the machine learning model, but we will attempt to draw some specific predictions from the model that suggest testable hypotheses for the psychological community. We also will have to deal with the well-known philosophical and psychological problem of distinguishing between cognition and metacognition. Finally, although we do not claim that the algorithms and representations used by the computational model actually exist within the head of an individual, IML theory provides an explicit computational model of metacognitive behavior and deliberate learning.

### 1.3 The Solution

The goal of integrating multiple learning algorithms is a daunting one, since it is an open question as how best to combine often conflicting learning-mechanisms. This research examines the metaphor of goal-driven planning as a tool for performing this integration. Learning is thus viewed as solving a planning problem (Cox & Ram, 1995; Hunter,

1990b; Ram & Hunter, 1992; Ram & Leake, 1995). By maintaining a declarative trace of reasoning that supports a particular choice of performance goals or plans, retrieving past cases of meta-reasoning that can explain the reasoning failure, and then directly inspecting and manipulating such explanations, an intelligent system can generate explicit learning goals that constitute desired knowledge changes.<sup>3</sup> A plan is subsequently assembled by choosing learning algorithms from the system's repertoire and ordering them in an appropriate way so that the learning goals are achieved.

The theory presented in this work is interesting because the choice of algorithm is not simply a function of the input, where the input is some set of assertions about the world, or even a faulty solution tree. Instead, since the input to the learner represents a trace or declarative representation of the prior reasoning that produced the solution, the choice of a learning strategy is a function of the reasoning that produced the error.<sup>4</sup> A solution plan is usually a structured set of physical operations that institutes changes in the world, such as chess moves that modify an external board position; in contrast, a reasoning trace is a structured set of mental operations that produces internal changes of mental states, selects problem operators, and eventually results in objects like solution plans. Thus, to decide on a choice of learning strategies, our theory of learning depends on introspection of the mental world, as much as it depends on an analysis of both the problem and the solution in the external world. In contrast, systems that make decisions based on a solution alone have only an indirect relationship to the actual causes of the failure.

This dissertation illustrates these problems and solutions within the context of a theory of introspective multistrategy learning with an implemented learning system called Meta-AQUA. The system learns by choosing a learning strategy on the basis of introspective explanations of its own performance failures. The performance task for Meta-AQUA is story understanding. That is, given a stream of concepts as the representation for a story sequence, the task is to create a causally connected conceptual interpretation of the story. As described in the previous section (on page 8), if the system fails at the task, its subsequent learning tasks are (1) *blame assignment* — explain the failure; (2) *decide what to learn* — form a set of explicit learning goals; and then (3) *learning-strategy construction* — construct a learning plan to achieve these goals.

---

3. Examples of learning goals are to answer a question or to reconcile two divergent assertions. Section 6.3 in Chapter VI will further enumerate the kinds of learning goals that exist in this theory.

4. Carbonell (1986) argues that an important insight into analogical reasoning is that solution derivations contain useful information beyond the information in the solution itself. His derivational analogy method is to map the derivation of old solutions onto new problems, rather than map old solutions into new solutions. This insight was one of the earliest arguments in favor of maintaining reasoning traces in support of learning.



As illustrated in Figure 8, the implementation of the solution to the above tasks has two parts. The system maintains a declarative trace of reasoning that leads to or supports a particular choice of goals or plans. Then, given a reasoning failure, the case-based reasoning (CBR) half of the learning subsystem retrieves past cases of introspective reasoning that support reflective blame assignment of the reasoning failure and that assist in the generation of a set of learning goals. Given such learning goals (representing desired changes in the system's background knowledge), the second part of the learning system is a nonlinear planner that constructs a partially ordered sequence of calls to different learning algorithms. Figure 7 on page 9 also shows this bipartite structure of the model.

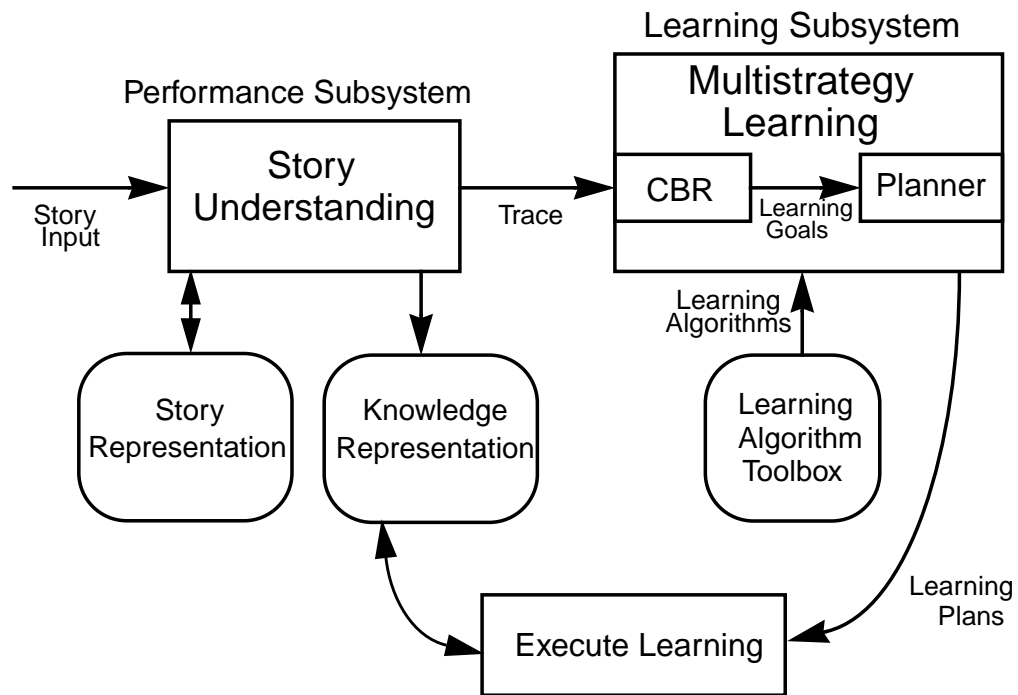


Figure 8. Basic Meta-AQUA system architecture

One of the key contributions of this thesis is a representational structure called a *meta-explanation pattern* (Meta-XP) (Cox, 1991; Cox & Ram, 1991; Ram & Cox, 1994). The structures can represent how an explanation is created (i.e., record the reasoning that generates an ordinary explanation), and they can represent causal patterns that explain why explanations fail (i.e., capture past cases of meta-reasoning about failure). In IML theory, the notion of a Meta-XP has been extended to represent performance failure in general, not just explanation failures.

A key construct in the solution to the strategy construction problem is the idea of a *learning goal* (Cox & Ram, 1994a; Ram, 1990; 1991; Ram & Hunter, 1992; Ram & Leake, 1995). Rather than specifying a desired state of the world, a learning goal represents a desired state of knowledge. A prototypical example of a learning goal is a question, the answer to which represents the achievement of the goal. A major hypothesis of this dissertation is that learning goals are *necessary* in order to mediate between the explanation of failure and the learning needed to avoid the failure; a direct mapping is not sufficient in all cases. Many case-based reasoning systems use the direct indexing of repairs by indexes that represent the conditions under which they are appropriate. This thesis will demonstrate that such linkage may lead to incorrect results when the chosen learning methods interact. Researchers cannot assume the learning algorithms are independent. Just as planning goals assist in alleviating the problems of interacting planning steps (i.e., painting a ladder and painting a ceiling; see Sussman, 1975 for an early discussion), learning goals can solve the problems of interacting learning strategies.

Although the Meta-AQUA system implements a performance module that conducts story understanding, the dissertation is not about story understanding or the comprehension process itself. The performance module is therefore quite simple. Likewise, although Meta-AQUA contains an indexed dynamic-memory module, no results will be reported on memory issues. Moreover, even the learning algorithms contained in Meta-AQUA's library are simplified reconstructions of well-known methods. This work does not contribute new learning methods *per se*; rather, it presents a new methodology for *dynamically combining* standard learning methods so that larger learning problems may be solved than those upon which individual learning algorithms were designed to operate.<sup>5</sup> The work is not concerned with first-order reasoning as much as it is concerned with the second-order reasoning required to learn deliberately. Thus, the research contributions are to be found in the representations used to model the mental world and the methods used to manipulate them, not in Meta-AQUA's first-order story-understanding performance.

## 1.4 Learning Goals and the Decomposition of the Problem

The research presented in this document arose from the pursuit of a simple question: Given a library of learning algorithms, how can one build an appropriate strategy with which to repair a system that has failed during its performance task? Originally, the task of answering the strategy-construction question appeared to be straightforward and rather direct. The goal of solving this problem became quite complex, however, because in order to answer the question a number of sub-questions arose. Each of these questions had to be

---

5. In a likewise reductionist fashion, Michalski and Ram (1995) propose a method for combining primitive inferential transmutations into standard learning methods.

answered in order to answer the main question, and many of these questions had further sub-questions that demanded answers.

### 1.4.1 The Learning Goal Metaphor

The pursuit of this research question also illustrates the notion of a learning goal, one of the key concepts of this dissertation. In its most basic form, the research question posed above is a goal to learn; this knowledge acquisition goal specifies a desired state of information to achieve in a scientific body of knowledge. Furthermore, the list of sub-questions generated by the original question forms a subgoal tree exactly like the goal trees generated by automated problem-solvers (see Figure 9, to which this chapter will return). Most importantly, however, this analogy between goal-driven problem-solving and learning is not only useful in understanding the thesis, but is a central analogy for the research itself.

Learning is like problem solving and planning. As such, learning indeed represents the proverbial search for knowledge. Moreover, to answer a question is to solve a problem, albeit an *internal* one; it is to achieve a new mental state that sufficiently fills a gap in a structured body of knowledge. One can externalize the new knowledge by formalizing it with mathematics, by writing it in English and by drawing a picture on a piece of paper, but the question's answer is essentially an abstract and internal informational-state, rather than a concrete state of the world, even when the answer can be composed as a list of physical operations in the world as can many plans.

Contrastingly, cynics will argue that the pursuit of research goals is basically an externalized problem to be solved and that the actual goal is to achieve a tangible such as a graduate degree. Any knowledge a student obtains is a collateral effect of the problem solving performed during the process while in school.<sup>6</sup> But there are many reasons to consider a learning goal an especially different and interesting type of goal instead.

The comparison between how a scientist or student conducts research and the view of learning as a purposeful pursuit of knowledge is more than just an interesting analogy. Although this thesis will not formulate a formal theory of scientific discovery (but see Nersessian, 1992; Thagard, 1993), it is nonetheless enlightening to consider the scientist's job and draw some conclusions. Knowledge is the central focus of both the scientist and

---

6. So, a continuum appears to exist upon which learning systems lie. At one end are theories like the one presented in this document in which goals are internal and learning explicit, and at the other end are theories in which goals are external and the learning implicit (or a collateral effect of processing external goals). An example of this second category is the learning theory embodied in Soar (Laird, Rosenbloom & Newell, 1986; Newell, 1990). See also Barsalou (1995).

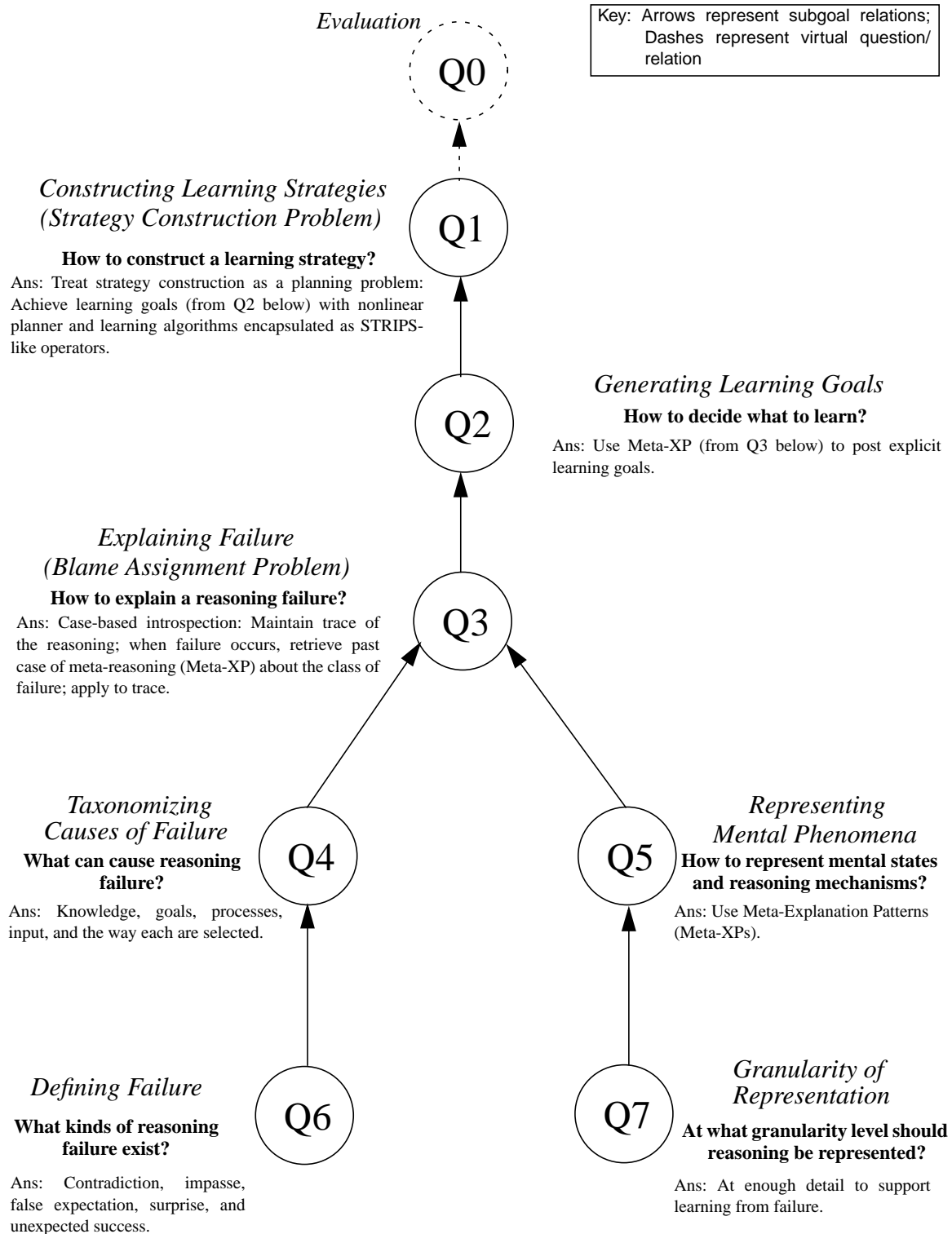


Figure 9. Primary research subgoal tree and contributions

the student. When scientists perform experiments, they are not just performing acts in the world in order to make certain states of the world become true; or if they do, these actions are subservient to the task of shedding light on the scientist's understanding (or lack thereof) about some facet of the world. The additional knowledge that can be gathered by an experiment (positive or negative) is the main reason for the actions, rather than the direct results of the experiments themselves. It is often not until after the experiment that this new knowledge is formalized in order to communicate it to the rest of the academic community. Scientific reasoning is also interesting because the scientist or student is reasoning about the thesis's line of reasoning, not just the natural world.

The answers to scientific questions are unusual as well. Scientists do not simply ask others if the answers they construct and the arguments they concoct are correct (although feedback from colleagues is useful). Answers to interesting scientific questions are not found by consulting the index of a textbook (although literature searches for related material are valuable). Granted, the pursuit of internal learning goals is a hybrid task, entailing both mental and physical actions. Physical actions are required to perform many of the information collecting tasks, much of the reasoning that scientists manipulate is in highly formal and external representations, and much of the logical reasoning has been formalized to a point that it can be carried out without much thought. But inevitably, when scientists are alone to think about their questions, theories, and inferences, they secretly peer into their own heads to consider how confident they are with certain conclusions, evidence and arguments. Introspection is fundamental in science because reasoning and knowledge are at the forefront. Learning in scientific discovery is a goal-driven cognitive behavior.

### 1.4.2 The Decomposition of a Learning Goal

Consider again the research goal tree of Figure 9 on page 14. Its function is to illustrate the central analogy of this dissertation and to serve as a pictorial guide when reading this thesis. Although the chronological pursuit of these goals was not in the order suggested by the numbering of the individual questions, the figure helps to visualize the logical structure of this dissertation. The central strategy-construction question, Q1, not only generates an entire tree of subgoals below it, but, once answered, raises the question Q0. The question of evaluation can be considered a supergoal if such a question is phrased as "How should one evaluate the method of strategy construction?" The figure also annotates each question with an abbreviated answer for each major research area (italicized in the figure) covered by this report. These answers constitute the major contributions of this research.

The original question (Q1) posed the learning-strategy construction problem. The answer to Q1 is to view learning as a planning problem. Given a learning goal, a learner can treat learning methods as problem-solving operators and use them to build a learning plan. This immediately raises the questions of what learning goals look like and how can such goals be generated (Q2). The answer to Q2 is that to generate the learning goals, the

learner needs to explain the failure. The learning goals originate from an explanation of the causes of the failure and are of multiple types as defined in a goal taxonomy. The blame assignment question (Q3) can then be answered if some representation exists of the prior reasoning and if the learner uses a characterization of the failure in order to retrieve some abstract explanation of the failure. But to represent failure, a reasoner must know what can cause failures (Q4) and it must have some formalism with which to perform the representation (Q5). Finally, to know the cause of the failure, the kinds of possible failures must be specified (Q6), and to represent the failure, a useful level of detail for the formalism must be determined (Q7). To be complete, IML theory must address all of these questions.

Traversing the tree from the bottom to the top, the following eight subsections will briefly describe the answers to these eight research questions in turn. Each subsection header is labeled with the corresponding question number from the figures in parentheses to organize the explanations. Subsequent chapters reexamine these eight questions in depth. By starting at the bottom of the tree and working upwards, readers will observe the support for answering a given research question before considering the question itself.

#### 1.4.2.1 What kinds of reasoning failure exist? (Q6)

This document will circumscribe failure with respect to two disparate but related types of reasoning processes. Reasoning may consist of problem-solving steps (such as planning, design, or troubleshooting), or it may involve comprehension of some stimulus (like understanding a story or results of a plan). A reasoning failure is defined as an outcome other than what is expected or a lack of some outcome, whether that outcome is a solution from some problem solving episode or an expectation from a comprehension process (Cox, 1993; Cox & Ram, 1994b). Five variations of failure exist under this definition.

If a system incorrectly analyzes some input, or solves some problem incorrectly, so that its expected analysis, prediction or solution differs from the actual outcome given some criteria or feedback, then a failure has occurred. This is the conventional notion of failure and will be termed a *contradiction*. Alternatively, an *impasse* is defined as either a failure of a process to produce any result or as the condition under which no process is available to attempt a result. Moreover, if a reasoner expects an event to occur, but nothing happens, then the failure is called a *false expectation*.<sup>7</sup> A false expectation will also be considered to exist when a unnecessary solution to a (non)problem is developed. If a system has no expectation, yet an event occurs which should have been expected, then a *surprise* exists (Cox, 1993). Finally, if a system expects that it will not be able to compute any answer or the correct answer, but it does nonetheless, then another failure class exists called an *unex-*

---

7. Note that Andrew's failure (see Figure 6 on page 6) is a false expectation, a type of error that AI systems seldom confront. Section 3.2.1.3 on page 47 provides more details.

*pected success*. This research defines reasoning failure in a larger scope than previous accounts (e.g., Hammond, 1989; Minton, 1988; Newell, 1990) and presents a unique declarative representation for each of the five classes of failure.

#### 1.4.2.2 What can cause reasoning failure? (Q4)

A taxonomy of general failure-causes answers Q4 by covering the possible causal factors in reasoning systems (Cox, 1992, 1993; Cox & Ram, 1994b; Ram, Cox, & Narayanan, 1995). Assuming that reasoning is the *goal-directed processing* of a given *input* using the reasoner's *knowledge*, only a limited number of classes of faults can be responsible for a given failure: the reasoning failure can originate in either the reasoner's goals, its performance strategies, the input, or the domain knowledge (see Table 1). Furthermore, given an additional assumption that knowledge is memory-based (i.e., subject to retrieval and organizational problems), then the organization of suspended goals (via indexes), processing strategy associations (via heuristics), or the organization of the domain knowledge (via indexes) may also be to blame. In the very last column, failures can be accounted for by attentional deficits that produce a flawed input context. In this causal taxonomy of reasoning failures, if one of these categories is responsible for an error, the item corresponding to the category is either absent or incorrect. If an item is correct, then that category contributes nothing to the failure. The taxonomy is comprehensive and goes beyond systems that limit the cause of error by assuming, for instance, noise-free input or other simplifications. The blame assignment task can therefore be characterized as a symptom-to-fault mapping from types of failure as specified in Section 1.4.2.1 to the causes of failure in Table 1.

Table 1: Basic taxonomy of causes of reasoning failure

	Domain Knowledge	Knowledge Selection	Goal Generation	Goal Selection	Processing Strategy	Strategy Selection	Input	Input Selection
<b>Absent</b>	Novel Situation	Missing Association	Missing Goal	Forgotten Goal	Missing Behavior	Missing Heuristic	Missing Input	Missing Context
<b>Wrong</b>	Incorrect Domain Knowledge	Erroneous Association	Poor Goal	Poor Selection	Flawed Behavior	Flawed Heuristic	Noise	Incorrect Context
<b>Right</b>	Correct Knowledge	Correct Association	Correct Goal	Correct Association	Correct Behavior	Correct Choice	Correct Input	Correct Context

### 1.4.2.3 At what level of granularity should reasoning be represented? (Q7)

If reasoning is to be represented declaratively so that the system can introspect upon it, then the answer to this question determines the level of abstraction at which the representations should be constructed. Schank, Goldman, Rieger, & Riesbeck (1972) claim that a mere set of two mental primitives (MTRANS and MBUILD) are sufficient to represent the utterances of humans concerning verbs of thought such as “I forgot that it was Sunday.” Alternatively, many in the AI community have built systems that record elaborate traces of reasoning, keep track of knowledge dependencies or inference, or encode much meta-knowledge concerning the structure of internal rules and defaults (e.g., Davis, 1980; Doyle, 1979). Our position is that the overhead involved with a complete trace of mental behavior and knowledge structures is intractable and does not reflect a reasonable capacity as possessed by humans. Instead, a system should be able to capture enough details to represent a common set of reasoning failures functionally necessary for learning (Cox, 1995). This document will explicitly represent all failure types enumerated in Section 1.4.2.1 with such a level of granularity and will specify what such representations offer an intelligent system.

### 1.4.2.4 How to represent mental states and reasoning mechanisms? (Q5)

By extending explanation pattern (XP) theory (Schank, 1986; Ram, 1989, 1991), the types of failures in Section 1.4.2.1 can be reasoned about deliberately. A meta-explanation pattern (Meta-XP) is an explanation of how and why an explanation goes awry in a reasoning system (Cox, 1991; Ram & Cox, 1994). Two classes of Meta-XPs facilitate a system’s ability to reason about itself and assist it in constructing a learning strategy. A *Trace Meta-XP* (TMXP) explains how a system generates an explanation about the world or itself, and an *Introspective Meta-XP* (IMXP) explains why the reasoning captured in a TMXP fails (Cox & Ram, 1992b). The TMXP records the structure of reasoning tasks and the reasons for processing decisions in a series of decide-compute nodes that resemble the derivational analogy traces of the PRODIGY system (Veloso & Carbonell, 1994). The IMXP is a general causal structure composed of primitive, network structures that represent typical patterns of reasoning failure. The IMXPs are retrieved and applied to instances of reasoning captured in TMXPs, and assist in forming the learning goals of the systems after failure occurs. Although the full details will be presented in later chapters, Figure 10 illustrates an instantiated IMXP bound to a (partially shown) TMXP reasoning trace that together represents a failure of forgetting to fill up with gas.<sup>8</sup> The failure symptom is a (memory) impasse as described in Section 1.4.2.1, whereas the failure fault is likely to be a missing index at the node I (i.e., likely to be the *missing association* cell of Table 1).

---

8. This might have been an explanation that Andrew considered while waiting for the bus to arrive. See Section 1.1.



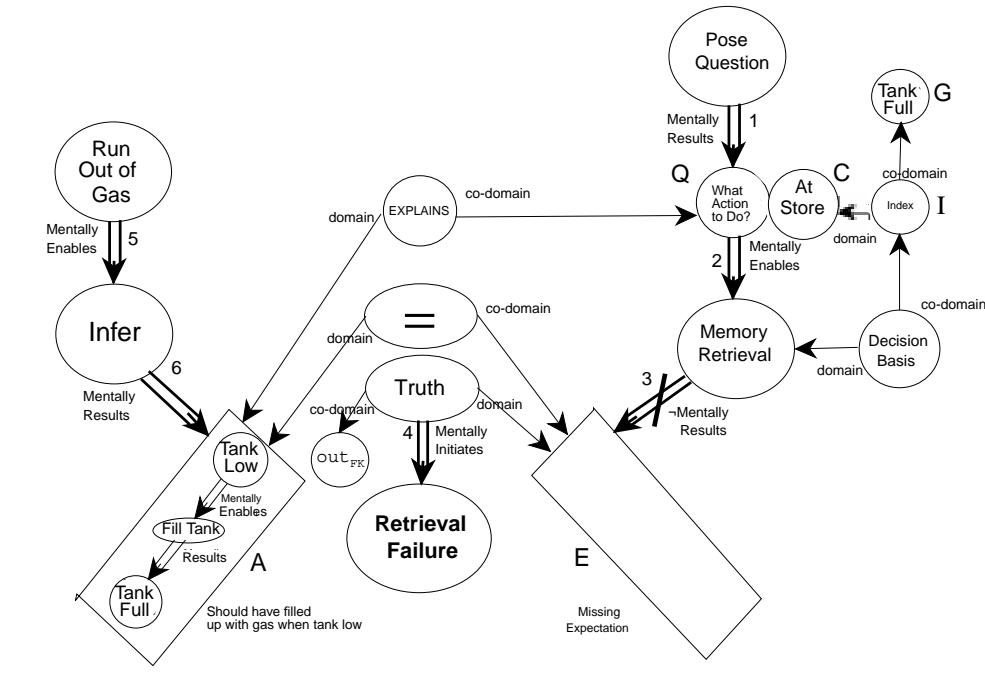


Figure 10. Forgetting to fill the tank with gas

A=actual intention; E=expectation; Q=question; C=context; I=index; G=goal

In addition to providing the basic representational framework of IML theory, this research has also generated two technical contributions. First, the formalization has made an explicit distinction between the background knowledge (BK) and the foreground knowledge (FK) of a reasoner. These divisions are necessary functional distinctions that allow a system to represent reasoning processes such as memory retrieval and are similar to psychological differences between long-term store and working memory. Secondly, to represent memory phenomena such as forgetting, it was necessary to extend Doyle's (1979) multi-valued logic. Instead of relying on values such as *in* the set of beliefs or *out* of the set of beliefs, Cox & Ram (1992a) extend them to cover *in* and *out* of the beliefs *with respect to a particular set of beliefs*. Thus, as seen in Figure 10, to represent forgetting one can mark a target memory item as being *out* of the set of beliefs with respect to the FK, yet *in* the set of beliefs with respect to the BK (Cox, 1994b; Cox & Ram, 1992a). This sufficiently represents retrieval failure while maintaining consistency.

#### 1.4.2.5 How to explain a reasoning failure? (Q3)

To reason effectively about one's own knowledge, goals, and reasoning requires an ability to introspect explicitly. A computational model of introspective learning is a second-order theory that contains a formal language for representing first-order processes and

that specifies the processing of instances of this representation. The reasoning algorithm used to perform such processing is similar to the algorithms used to reason about events and processes represented in the original domain, that is, case-based reasoning algorithms. Case-based understanding (1) takes as input some event in its domain along with its context, (2) based on salient cues in the input, retrieves a prior case to interpret the input, then (3) adapts the old solution to fit the current situation, and finally (4) outputs the result as its understanding of the domain. Similarly, *case-based introspection* (Cox, 1994a) (1') takes as input a representation of some prior faulty reasoning in the form of a TMXP, (2') based on salient cues in the input, retrieves a prior case of reflection in the form of an IMXP to interpret the input, then (3') adapts the old case to fit the current situation, and finally (4') outputs the result as its self-understanding and learning goals. Here, the system's domain is itself.

Case-based introspection has proven useful during blame-assignment in Meta-AQUA. Failure analyses cannot always look to the external world for causes. Often the assignment of blame is with the knowledge and reasoning of the system itself. Therefore, when Meta-AQUA encounters a reasoning failure while reading stories, it uses case-based introspection to explain why it failed at its reasoning task. The system uses this analysis as a basis to form learning goals and subsequently to construct a learning plan to repair its memory. The control algorithm used for introspective (second-order) reasoning is essentially the same as the XP-application control algorithm used in explanatory (first-order) reasoning in AQUA (Ram, 1989, 1991, 1993, 1994) and SWALE (Kass, Leake, & Owens, 1986; Schank & Leake, 1990).

#### 1.4.2.6 How to decide what to learn? (Q2)

Once a failure is understood and its causes identified, learning must be focussed by deciding on a number of specific targets. Learning goals represent these desires explicitly (Cox & Ram, 1994a; Ram, 1990, 1991; Ram & Hunter, 1992; Ram & Leake, 1995). The learning goals are designed so that, if achieved, they will reduce the likelihood of repeating the failure. After blame assignment, the learning goals are obtained from the instantiated IMXP that is bound to the trace of reasoning represented in the TMXP. The IMXP contains a list of learning goals that point to the most likely sources of error in the graph structure representing the pattern of reasoning failure. Some learning goals seek to add, delete, generalize or specialize some concept or procedure. Others deal with the ontology of the knowledge, that is, with the kinds of categories that constitute particular concepts.

Many learning goals are unary in that they take a single target as argument. For example, a *knowledge acquisition goal* (Hunter, 1990b; Ram, 1990, 1991) seeks to determine a single piece of missing knowledge, such as the answer to a particular question. A *knowledge refinement goal* seeks a more specialized interpretation for a given concept in memory, whereas a *knowledge expansion goal* seeks a broader interpretation that explores connec-

tions with related concepts. Other learning goals take multiple arguments. For instance, a *knowledge differentiation goal* (Cox & Ram, 1995) is a goal to determine a change in a body of knowledge such that two items are separated conceptually. In contrast, a *knowledge reconciliation goal* (Cox & Ram, 1995) is one that seeks to merge two items that were mistakenly considered separate entities. Both expansion goals and reconciliation goals may include or spawn a *knowledge organization goal* (Ram, 1993) that seeks to reorganize the existing knowledge so that it is made available to the reasoner at the appropriate time, as well as modify the structure or content of a concept itself. Such reorganization of knowledge affects the conditions under which a particular piece of knowledge is retrieved or the kinds of indexes associated with an item in memory.

#### 1.4.2.7 How to choose or construct a learning strategy? (Q1)

Finally, given a set of learning goals, a decision must be made to determine which learning strategies are most appropriate for achieving it. The approach taken is to treat the learning task as a traditional planning problem, creating a learning plan that is composed of a series of learning algorithm calls that will achieve the learning goals. However, unlike learning algorithms executed by single-strategy systems, the learner must dynamically consider possible interactions that may occur between the learning strategies. It is therefore important to recognize that when multiple items are learned from a single episode, the changes resulting from one learning algorithm may affect the knowledge structures used by another algorithm. Such dependencies destroy any implicit assumption of independence built into a particular learning algorithm used in isolation. For example, if one algorithm generalizes a conceptual definition, thus introducing or altering constraints on an attribute of the definition, any memory re-indexing based on this attribute must occur after the modification, rather than before it, in order for the indexing to be effective.

A standard nonlinear planner is therefore used to resolve these types of dependencies and goal interactions (Cox & Ram, 1995). The planner is treated as a black box. It is provided with an input of specific learning goals and the context in a predicate representation. The learning algorithms are represented in the form of standard STRIPS (Fikes & Nilsson, 1971) operators so that the planner can reason about these interactions, pick the appropriate algorithms, and sequence the algorithm calls as partially ordered steps in a learning plan or strategy. The algorithms can then be executed in the sequence specified by the plan.

#### 1.4.2.8 How can the research be evaluated? (Q0)

The theory developed during this research has been evaluated from both machine learning and cognitive science perspectives. To investigate and evaluate the theoretical problems, an introspective version of AQUA<sup>9</sup> called Meta-AQUA was implemented. AQUA is a question-driven story understanding system that learns about terrorist activities. Its performance task is to “understand” the story by building causal explanations that link the individual events into a coherent whole. Meta-AQUA adds introspective reasoning and

learning using Meta-XP structures. Meta-AQUA's performance domain consists of using knowledge of terrorist activities (taken from the original AQUA system) to understand and explain stories of drug smuggling. Meta-AQUA's learning domain consists of using knowledge of reasoning failure (contained in the Meta-XP structures) to understand, explain, and learn from its own errors of story understanding. In support of the main system, a large frame system manages Meta-AQUA's knowledge representation. A publicly available story-generation system called Tale-Spin (Meehan, 1981) supplies automatically-generated input at Meta-AQUA's front end, while a publicly available nonlinear planning system called Nonlin (Ghosh, Hendler, Kambhampati, & Kettler, 1992; Tate, 1976) generates the final learning plan at Meta-AQUA's back-end.

We evaluate the program in a number of contexts. In machine-learning environments, the emphasis has been on the problem of choosing computational learning algorithms, given some learning task. Hand-coded examples represent a number of paradigmatic cases of reasoning failure (see Section 2.1 for two such examples). The examples demonstrate the utility of our approach when learning algorithms interact. Also, a series of experiments with the Tale-Spin example generator empirically investigate the performance features of the system with and without interactions (see Section 9.2 for the experimental results). In this study, Meta-AQUA performed better in a fully introspective mode than in a reflexive mode in which learning goals were ablated. In particular, the results lead to the conclusion that the deciding to learn stage that posts learning goals is a necessary stage if negative interactions between learning methods are to be avoided and if learning is to remain effective. In addition, the IMXPs have been shown to apply without modification in a second story-understanding domain (Cox & Freed, 1994).

With respect to a cognitive science evaluation, the emphasis has been to derive a plausible model of how human learners choose particular approaches or metacognitive strategies given some problem-solving task.<sup>10</sup> To test its plausibility, Meta-AQUA was modified in order to model one protocol in a set of human data in a LISP troubleshooting domain (Cox & Kell, 1993). The protocol was chosen from data gathered in the School of Education at Berkeley concerning the behavior of novice LISP programmers. These data support the positive relationship between metacognitive reasoning and learning in novel problem-solving domains (Pirolli & Recker, 1994). These data were collected and analyzed without knowledge of this dissertation work, and the Meta-AQUA system was developed without knowledge of the data. The results support the claim that the theory is a reasonable and sufficient model of reflection and learning and that the theory is appropriate for both problem-solving and story-understanding tasks.

---

9. AQUA stands for Asking Questions and Understanding Answers.

10. See also Cox (1994c) for a description of the relation between IML theory and research into the roles of metacognition, problem solving and cognitive aging in human subjects.

The theory has also been used to model human data in real-world problem-solving tasks. In collaboration with colleagues from the Industrial and Systems Engineering Department of the College of Engineering at Georgia Tech, we have successfully used this approach in the domain of diagnostic repair of circuit boards, modeling the behavior of expert troubleshooters at NCR's electronics assembly plant in Atlanta (Ram, Narayanan, & Cox, 1995). The Meta-TS system was developed as a dual model. Based also on IML theory, it relies on shallow associative knowledge of failure, rather than deeper causal knowledge of failure (as is the case with the Meta-AQUA system).

## 1.5 Overview of the Dissertation

This dissertation follows the structure of the goal tree decomposition as seen in Figure 9 (p. 14). The questions in the tree are roughly divided into two parts. The lower part (i.e., questions Q4 through Q7) contains questions of representation and content, whereas the upper part (i.e., questions Q1 through Q3) pertains to process and functionality. After concluding Part One, "PRELIMINARIES," with a chapter that introduces the notion of content theories and process theories, the thesis will continue with Part Two, "A CONTENT THEORY OF MENTAL REPRESENTATION," followed by Part Three, "A PROCESS THEORY OF LEARNING AND INTROSPECTION."

Part Two presents a knowledge-level theory of the content of the mental world. The effort invested into the construction of this theory was substantial, and thus, the representational issues will dominate a large section of the initial part of the thesis. The results, however, provide foundational support for the remainder of the thesis. Part Three explains how a system can perform particular learning functions and improve its performance. The resulting answers will constitute a process theory of introspective learning. This section will present the major computational steps necessary to create a learning strategy and with which to reason about the internal world of mental representations as provided by part one, the content theory.

Part Four concludes the thesis by first examining the Meta-AQUA implementation of IML theory. The architecture of the system is explained and the implementation is evaluated with respect to how well it has answered the questions in Figure 9. As with any good question, this research has spawned more questions than answers, so Part Four also addresses future research, as well as related research. A concluding chapter highlights the main points of the thesis, and an epilogue openly speculates on the implications of this research.



## CHAPTER II

### CONTENT THEORIES AND PROCESS THEORIES

*By distinguishing sharply between the knowledge level and the symbol level the theory implies an equally sharp distinction between the knowledge required to solve a problem and the processing required to bring that knowledge to bear in real time and real space.*

—Alan Newell (1982), p. 117.

Two parts or sub-theories exist within any complete cognitive theory that claims to explain, describe, or predict intelligent behavior and reasoning. The *content theory* provides the vocabulary and structure for representing knowledge, as well as the ontology and content of the knowledge. Content theories provide a component theory that specifies the objects or components in the domain and the features that best describe the components. Also, a content theory provides constraints and inferential relationships between the features. Content theories therefore possess commitments to both domain ontology as well as domain physics in a body of knowledge (Domeshek, 1992). The *process theory* specifies the classes of transformations performed on such knowledge (Birnbaum, 1986; Domeshek, 1992). Moreover, a process theory is a *functional theory* if the processes are justified by some teleological commitment (i.e., if the theory defines a specific functional role for each process that contributes to the cognitive task for which the theory is offered as an explanation). Because the focus of this research is reasoning about reasoning failure (in order to learn), rather than reasoning about some external task, our process theory is a description of second-order introspective processes in the learning task, as well as the first-order processes in the performance task. Furthermore, to learn effectively from reasoning failure, the learner must be able to represent the cognitive processes responsible for failure explicitly. Our content theory is thus unique in that it becomes a descriptive language to explicitly represent both the first-order processes described in our process theory, as well as the events in the external world.

The intent of this thesis is to outline a broad theory of introspection, understanding, and learning by providing specific commitments as to the kind of processes that account for such cognitive activities and the kind of representational language that is suited for compu-

tationally describing and making inferences from these phenomena. To make clear the first-order processes that need to be captured by the representations, to clarify the special relationship between the content theory and the process theory, to foreshadow the implementation of these theories in the Meta-AQUA multistrategy learning system, and to provide concrete examples that will support assertions throughout the remainder of the dissertation, the following section (Section 2.1) will describe two of the short, hand-coded stories Meta-AQUA understands and from which it learns. The subsequent section (Section 2.2) discusses the distinction between content theories and process theories in terms of the difference between knowledge and process. The third section (Section 2.3) then relates these concepts to both the task and domain of Meta-AQUA and to the two implementational examples.

## 2.1 The Drug-Bust Examples

As previously described (Section 1.4.2.8), the performance task of Meta-AQUA is to understand stories in the domain of drug-smuggling, given its past experience with terrorist stories. From a conceptual representation of the input sentences, the story understanding task is to build a coherent interpretation of such input using the knowledge structures in its memory stores. Its memory is divided into a foreground knowledge (FK), where it maintains the current model of the story, and a background knowledge (BK), where the system stores a library of declarative knowledge structures including explanations, cases, and representations of its own reasoning processes. When Meta-AQUA detects anomalies or other interesting input in the story, it attempts to explain the anomaly; otherwise, it skims the story by applying scripts (Cullingford, 1978; Schank & Abelson, 1977). If a failure of explanation occurs, the system must explain the failure, decide what to learn, assemble a learning strategy, and execute that strategy.

### 2.1.1 A Common Contradiction

As an example, consider the simple story in Figure 11. Given the drug-bust story, the system attempts to understand each sentence by incorporating it into its current story representation. Numerous inferences can be made from this story, many of which may be incorrect.

In the story, sentence S1 produces no inferences other than that sniffing is a normal event in the life of a dog. However, S2 produces an anomaly because the system's definition of "bark" specifies that the object of a bark must be animate. The program (incorrectly) believes that dogs bark only when threatened by animate objects. Since luggage is inanimate, there is a conflict. This anomaly causes Meta-AQUA to ask itself why the dog barked at an inanimate object. Given a prior explanation about dogs barking when threatened by persons, it hypothesizes that the luggage somehow threatened the dog. It suspends the



S1:A police dog sniffed at a passenger's luggage in the airport terminal.  
 S2:The dog suddenly began to bark at the luggage.  
 S3:The authorities arrested the passenger, charging him with smuggling drugs.  
 S4:The dog barked because it detected two kilograms of marijuana in the luggage.

---

Figure 11. Hand-coded story HC1 (from Cox & Ram, 1991)

question, however, after it no longer can proceed due to the lack of additional information. S3 posits an arrest scene that reminds Meta-AQUA of an incident in which weapons were smuggled by terrorists; however, the sentence generates no new inferences concerning the previous anomaly. Finally, S4 causes the original question generated by S2, “Why did the dog bark at the luggage?” to be retrieved. Instead of revealing the anticipated threatening situation, however, S4 offers another hypothesis: “The dog detected drugs in the luggage.”

At this point, the system has detected an explanation failure, and so it suspends the performance task. Until now, all processing was first-order reasoning about the story using first-order knowledge about the domain of criminal activities. Learning involves second-order reasoning about the prior, faulty story-understanding effort using second-order knowledge about failures and about the processes in the first-order task. Introspective learning must be able to represent the processes that detect the anomalies in the story, that generate explanations, and that verify the explanations once made.

Meta-AQUA uses a case-based approach to explain its reasoning failures (i.e., perform blame assignment). The system characterizes the reasoning error as an expectation failure caused by the incorrect retrieval of a known explanation (“dogs bark when threatened by objects,” erroneously assumed to be applicable), and a missing explanation (“the dog barked because it detected marijuana,” the correct explanation in this case). During blame assignment, Meta-AQUA uses this characterization as an index to retrieve an abstract Meta-XP (IMXP) that is applied to a trace of the reasoning (TMXP) that produced the failure. This structure then aids the system in posting a number of learning goals that, if achieved, will modify the system’s BK so that similar errors are not repeated in future episodes. The modifications are a change of the dog-barking definition to remove the `animate-object` constraint, a generalization of the new explanation, and a mutual reindexing of the new explanation with respect to the erroneous threaten explanation.

The explanation failure was a common occurrence when learning about a new domain. When a concept is being learned, it may be overly specialized. Slight variation on the concept will cause the system to try to explain it, but without experience with the concept, the system may generate an inappropriate explanation. The proper explanation may not be known because the situation is novel. Much of the power of the IML method comes from a library of such common patterns of reasoning failure.

### 2.1.2 A Baffling Situation

After processing the previous story, Meta-AQUA's BK contains two explanations for why dogs bark: the memory contains an explanation for dogs that bark when threatened (indexed by `dog-barks-at-animate-object`) as well as the explanation for dogs that bark because they detect contraband (indexed by `dog-barks-at-container`).<sup>11</sup> Meta-AQUA is then given a second story (Figure 12).

S1: The police officer and his dog entered a suspect's house.  
 S2: The dog barked at a pile of dirty clothes.  
 S3: The police officer looked under the clothes.  
 S4: He confiscated a large bag of marijuana.  
 S5: The dog was praised for barking at the occluding object.

---

Figure 12. Hand-coded story HC2 (from Cox, 1994b)

Although the initial sentence, S1, causes no unusual processing, the second sentence, S2, is interesting to Meta-AQUA because the system has recently changed its concept of `dog-bark`. The system therefore poses a question to ascertain the reason the dog barked. Unfortunately, because it is barking at neither an animate object nor a container, no XP is retrieved to produce a cause for the event. The question-answering process is subsequently suspended because of the impasse, and the question is indexed in memory. Meta-AQUA uses an opportunistic strategy of waiting until the story provides further information before resuming the process.

Sentence S3 causes the system to postulate a possible causal link between S2 and S3 simply because of their temporal relation; however, no evidence directly supports their

---

11. Section 8.4.2, "Indexing," starting on page 192, describes the implementation of the indexing scheme used to generate indexes such as these.

association. S4 reminds the system of a case in which contraband was confiscated. The system thus infers that the suspect was most likely arrested. Finally, S5 causes a reminding of the earlier question about the dog barking at the pile of laundry. The reasoning that was associated with this previous question is resumed. The system also infers a causal relation from S5. That is, although the sentence does not explicitly assert it, Meta-AQUA concludes that the dog's detection of the marijuana caused the dog to bark in the first place. As a result, this conclusion answers the original query.

Reviewing the trace of processing that led up to this conclusion, Meta-AQUA characterizes its condition as being “baffled;” that is, it could not explain why the dog barked and instead just “drew a blank,” and now it has inferred one. The system retrieves an IMXP based on this characterization, which again helps it explain its reasoning failure. The IMXP is a declarative representation of memory retrieval failure. The system is not able to determine *a priori* whether an explanation actually existed in memory that it could not previously recall, or whether it lacks the knowledge with which it could have produced the explanation. It thus poses an introspective question about its own IMXP, “Does such an explanation exist in memory or not?”

The answer to this question is obtained by going ahead and performing a generalization on the inferred explanation (producing the XP “dogs generally bark when detecting contraband”), indexing it by the context in which the system inferred the explanation (“dogs barking at piles of objects”), and then watching for a similar explanation in memory when it stores it. It thus finds at storage time the explanation produced by the previous story (from Section 2.1.1) and must backup from the strategy the system had originally intended. So now the system generalizes the two explanations with respect to each other. It thus produces a better explanation than either the inferred one or the one from the previous story: dogs bark at objects that hide contraband, not simply at containers. So that these types of explanations will not be forgotten again, it indexes the new explanation by potential hiding places.

Like the “common contradiction” example, this second example illustrates another typical explanation failure. When novices are learning about new phenomena, they often forget the explanations generated by previous experiences. It takes a few times to see a new behavior before the learner understands the purpose of the behavior and conditions under which explanations of the behavior apply. It is these kinds of abstract patterns of failure (contradictions and baffling situations) that comprise the integral pieces of knowledge that a content theory must represent and with which the cognitive processes involved in introspective learning will use to construct a learning strategy.

## 2.2 Knowledge and Process

Although differing in technical terminology, many researchers have made the distinction between knowledge and process in terms of the division between representation and the transformations on such representations. Notwithstanding the insights of previous theories, we claim that a content theory is not merely a logical description of the domain under consideration, and the process theory is not simply an enumeration of the kinds of inferences that can occur in such domains. Instead, the theories provide a declarative representation of those aspects of the domain that are salient and teleologically useful to process transformations and a vocabulary with which to express such representations. Moreover, in an introspective theory of learning, the content of knowledge includes a declarative representation of the cognitive processes themselves because, as illustrated in the previous section, a learner must be able to represent and explain how processes fail, if it is to learn from its mistakes. In IML theory, both the content theory and the process theory have two parts: One part explains cognition in the performance task (story understanding) and the other part explains cognition in the introspective task (learning).

The division between knowledge and process is a common one. In computer science, a significant division between data models and algorithms exists, both of which are considered fundamental to a principled understanding of computation (Aho & Ullman, 1992). A data model is the abstract representation of objects and operations, whereas, algorithms represent structured specific computational details for manipulating these data. For example, an array is a data model of linear sequences of like elements. The operations consist of functions to access or store a given element. Algorithms exist to sort the elements in an array. This separation is much like the division between knowledge and process (inference) in artificial intelligence. There is a difference between the representation of the objects and events (operations between objects) and the processes that operate on these representations.

Newell (1982) made a similar distinction when separating knowledge-level theories and symbol-level theories. At the knowledge level, agents make decisions according to the principle of rationality. They act when they possess knowledge that such actions will achieve their goals. However, the knowledge that agents use to determine what action to follow can be separated from the process that is used to actually determine such actions.<sup>12</sup> Thus, all processes exist on the symbol level that is a lower level of abstraction. Only at the symbol level do knowledge-level abstractions assume a computational reification and specification.

McCarthy & Hayes (1969) used this same division when speaking of the difference between epistemology and heuristics; that is, between the representation of the knowledge and inferences used with such knowledge, and the implementational details used to instantiate such representations and inferences. This is also reminiscent of the philosophical dis-

inction (Ryle, 1949) between “knowledge that” (i.e., declarative knowledge in AI terms) and “knowledge how” (i.e., procedural knowledge). However, with the case of the Meta-AQUA examples, the relevant division is a peculiar one and the separation not as clear. The reason for this condition is that, because the system uses a second-order introspective process to learn about the first-order reasoning processes, the critical data is a representation of the first-order processes themselves. In the theory we describe in this thesis, both the content and process theories have first-order and second-order components. Although both the representations and the processes are thus convoluted, a major goal of this thesis is to unravel the content and process descriptions in a comprehensible fashion.

In any case, the representation of knowledge is a very difficult task, even in first-order cognitive theories. Early work in logic demonstrated that some peculiar problems exist when representing knowledge in a general manner (Moore, 1977). Foremost, the term “to know” cannot be treated as a standard logical predicate of the form  $\text{Know}(\text{John}, P)$ . In ordinary logic, one can substitute inner terms that are equivalent in truth without changing the overall truth values of the outer expressions. This property is called *referential transparency*. Thus, if both A and B are true, substituting any true term for either A or B will not change the truth value of the expression  $A \wedge B$  itself. However, the second term of the above predicate  $\text{Know}$  is referentially opaque. For example, it may be true that “If it rains, then John’s car will get wet.” Now when given that it is raining, it is necessarily true that John’s car will get wet. But alternatively, if it is true that “John knows that if it rains, then his car will get wet” and it is also true that it is raining, one cannot necessarily infer that John knows that his car is wet. This disparity is equivalent to the logically correct sequence  $A \rightarrow B; A, \text{therefore } B$  versus the incorrect inference  $\text{Know}(\text{John}, A \rightarrow B); A, \text{therefore } \text{Know}(\text{John}, B)$ .

Another important contribution of the logic community is their early emphasis upon declarative representation (see the discussion in Birnbaum, 1991). But for the logicians, building a representation means to design logical inference mechanisms or axiomatizations for particular verbs or actions such as “to use” (McCarthy & Hayes, 1969) or “to know” (Moore, 1977). They worry about the syntax of well-formed formulae (e.g., the constraints

---

12. Note that this assertion is not without its critics. Palmer (1978) argues that knowledge and process are interrelated because the representation is dependent upon the purpose for which it is used. This functional dependence is also echoed by Schank, Collins & Hunter (1986) in the context of inductive-category formation. Although for the purpose of engineering a theory, we claim that the separation has its benefits when considering knowledge in an quasi-independent fashion, we also recognize the intertwined relations between the two components. These relationships will be made explicit where possible. See, for example, the discussions in Section 2.3. Note, however, that strong proponents of the separation of knowledge and process do exist, such as Tulving (1994) who claims that “there is no direct correlation between kinds of knowledge and forms of knowing, between representation and process.” (p. vii)

on logical connectives), the semantics of correspondence (how terms can be mapped to the real-world or possible worlds), and above all else, absolute consistency. Although gaining logical precision with such an agenda, they pay the price of painstaking expressiveness and brittleness given the need to avoid inconsistency at all costs. When it comes to representing concrete objects and events in particular domains with particular tasks they have less to say.<sup>13</sup> Moreover, within the predicate logic, formulae have the aforementioned property of referential transparency. Thus, logicians perhaps are tempted to ignore the representation of terms that refer to objects and events simply because, as long as the truth value remains constant, it does not matter what the content of the term may be.

Alternatively, when a researcher wishes to build a content and process theory of representation, the individual begins with the domain. Analysis of the domain determines the significant processes within that domain and those features of the domain that must be represented to support these processes. The content representation provides an ontological vocabulary of terms with which to signify the meanings, the relations between the terms, and a syntax for combining the terms and making inferences from them. Indexing vocabulary specifies those features under which representations are stored and retrieved from memory (see Birnbaum, 1989). The process theory provides a functional account of those cognitive processes that produce the behavior in the domain. The overarching goal is to provide a interlocking language for representing concrete experiences and behavior, rather than logical assumptions or deductions. The most interesting challenge of IML theory is not just to produce a process and a content theory concerning the task of story understanding in a domain of criminal activities, but to produce two additional theories that apply to the task of introspective learning in the domain of story-understanding failures.

## 2.3 The Domain of Story-Understanding Failures

A typical cognitive theory accounts for a specific class of intelligent tasks in a particular domain of effort requiring reason. With respect to the first-order task of story understanding and the domain of drug-smuggling from the Meta-AQUA examples (Section 2.1), a content theory provides a language that adequately describes specific objects and events in the world of smuggling and general planning for criminal activity; whereas, a process theory specifies the mental processes involved in story understanding such objects, events, and plans. That is, the content theory of story understanding is a first-order theory of how humans mentally represent the important events and characteristics of the story when read-

---

13. But as exceptions, see the work of Hayes (1979/1992) and then ARPA's knowledge sharing effort that uses the predicate logic as a starting point from which to formalize a number of domains (e.g., Gruber, 1993; Patil, Fikes, Patel-Schneider, McKay, Finin, Gruber, & Neches, 1992). An internet URL on the subject is <http://www-ksl.stanford.edu/kst/kst-overview.html>.

ing, and the process theory describes the important mental manipulations of these representations that produce a coherent interpretation of the story. The process theory of understanding explains and predicts the behavior of agents engaged in reading about criminal behavior, such as inferring goals and plans of the actors involved in the story (described with the content theory), and incorporating such inferences into the overall interpretation. When explaining unusual events in a given story (such as dogs barking at inanimate objects) the process theory enumerates the kinds of reasoning performed by the reader when given a representation of the prior events in the story contained in the FK and the general and specific knowledge in the reader's BK. Moreover, the theory describes the transformations necessary to generate an explanation and would specify the connectivity between cooperating processes. The content theory of story understanding describes those features and relationships of the domain in need of representation<sup>14</sup> and enumerates a vocabulary with which to express such representations. The objects in the content theory (domain knowledge of story understanding and smuggling) and processes in the process theory (the transformations on such knowledge) are thus related, but mostly distinct.

As shown in Figure 13, when reasoning about a story, the reader develops representations for the events that produce state changes in the characters and objects of the story. When explaining a novel or unusual action in the story, the reasoner performs mental actions or events that produce new interpretations of these representations. Note that both **Story-Repr1** and **Story-Repr2** are mental representations for the state changes in the story. For example, **Story-Repr1** might be the representation for the dog barking at the luggage, whereas **Story-Repr2** might be a modified representation explaining why the dog did such an act. The content theory of story understanding provides the language for these representations (e.g., scripts, cases, and XPs), while the process theory of story understanding presents a description of the processes that transform them (e.g., script processing, anomaly detection, and explanation).

When adding a second-order theory of introspective learning, however, the content and process theories become more intimately related. The content theory of introspective learning must be able to represent the events and state changes that the process theory of story understanding describes. The process theory of introspective learning is a theory of how these second-order representations are changed. Now, if the system is to process memories of its own processing, then a language is needed with which to represent the processing itself. During reflection, the processes transform and operate upon descriptions of themselves. So, because the external domain of this thesis is story understanding, the process theory must specify the cognitive processes that account for understanding and expla-

---

14. Equally important, a content theory implicitly, and sometimes explicitly, determines those features and relationships which are *not* worth representing. It therefore provides a computationally tractable level of abstraction, rather than an exhaustive, descriptive inventory of the object domain.

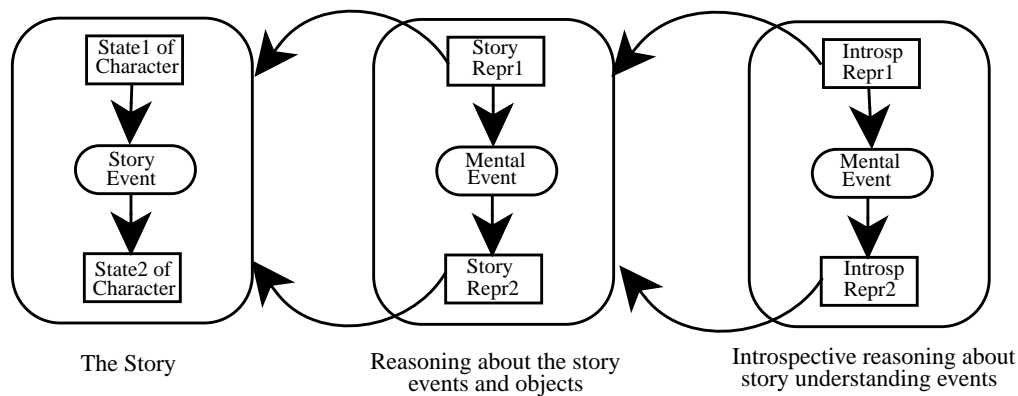


Figure 13. Multilevel representations and processes

nation (as would any standard theory of understanding). In addition, it must be able to represent those features crucial to improving the performance. In a story understanding system that learns, it must therefore be able to declaratively represent failure and how the system responds to failure (i.e., those features most important to improving the performance of the system). But, moreover, the content theory of introspective learning concentrates on declarative representations of these comprehension processes; the content theory of criminal agents and events is secondary.

Again looking at Figure 13, when reasoning about a explanation failure, the learner develops representations for the mental events that produce state changes in the interpretations of the story. When explaining a reasoning failure, the reasoner performs mental actions or events that produce reasons for the failure. Note that both *Introspr-Repr1* and *Introspr-Repr2* are mental representations for the state changes in the story-understanding process. For example, *Introspr-Repr1* might be a trace of the reasoning that produced the conclusion that the dog barked because it was threatened, whereas *Introspr-Repr2* might be an introspective explanation for why the reasoning in *Introspr-Repr1* failed. The content theory of introspective learning provides the language for these representations (e.g., TMXPs and IMXPs), while the process theory of learning presents a description of the processes that construct a learning strategy.

The process theory within the IML framework contends that the performance task of story understanding consists of those processes depicted in Figure 14. An understanding goal is input into an analysis process that determines whether anything unusual exists within the story input. If so, it passes the unusual input to the next phase for further processing; otherwise, it skims the input. The anomaly is given to the explanation generation process, which finds a relevant explanation strategy from the system's memory. This pro-



cess then generates an explanation and passes it to a verification phase. The explanation, along with a goodness of fit,<sup>15</sup> is then returned as a result, or, if the plan is insufficient, the problem is suspended and the process restarted when additional information is present.

The content theory of story understanding provides the vocabulary used to describe stories of drug-smuggling and terrorist activity as perceived by a reader. It contains both domain-independent information (such as the facts that stories have main actors and events have results and preconditions) and domain-dependent facts (such as the typical goals and plans of persons who use coercion and stealth) and other causal features and relations relevant to understanding actors and actions in the stories. To reason explicitly about the process of explanation and story understanding, however, a system must be able to represent not only the final result of comprehending the story, but it must also possess a way of recording a trace of the processes that produces the explanation. It is not sufficient to simply annotate the final explanations with features signifying what occurred during the understanding process.<sup>16</sup> Thus, a content theory of introspective learning (i.e., of story-understanding failures, or more generally, of reasoning failure) provides the vocabulary used to describe these traces and the representations of knowledge used to explain process failures.

Instead of simple annotations, it is desirable to create a chain of structures or nodes, one for each process in the planning effort. Each node records the input and output, the bases and context for its results, and a link to the following process (details are presented in Section 4.4.1). In this way the system can represent, for example, an anomaly analysis, an explanation generation, and a verification, producing an explanation that did not work, then a reformulation of the question followed by another series of analyze, generate and verify steps. A benefit of producing this record is that it is also available for use by subsequent processes in the planning mechanism. By recording the explanation process and representing it explicitly, far more information is available with which to understand the current story, as well as to improve interpretation of future stories.

The approach this document will take, then, is consistent with the above analysis. It will develop a specific model of reasoning, along with a representational language and a knowledge taxonomy for expressing instances of reasoning and reasoning failure. Once expressed in some declarative, inspectable form, a system can process instances of its own

---

15. The implementation actually returns either verified or not verified, rather than a qualitative fit.

16. One reason for precluding such an approach is that the process of explanation is recursive; that is, an explanation may be generated by an arbitrary number of passes through the explanatory loop. For example, an explanation may have preconditions that themselves require explaining. An annotated explanation is therefore insufficient to distinguish between the various activities at similar points in the process.

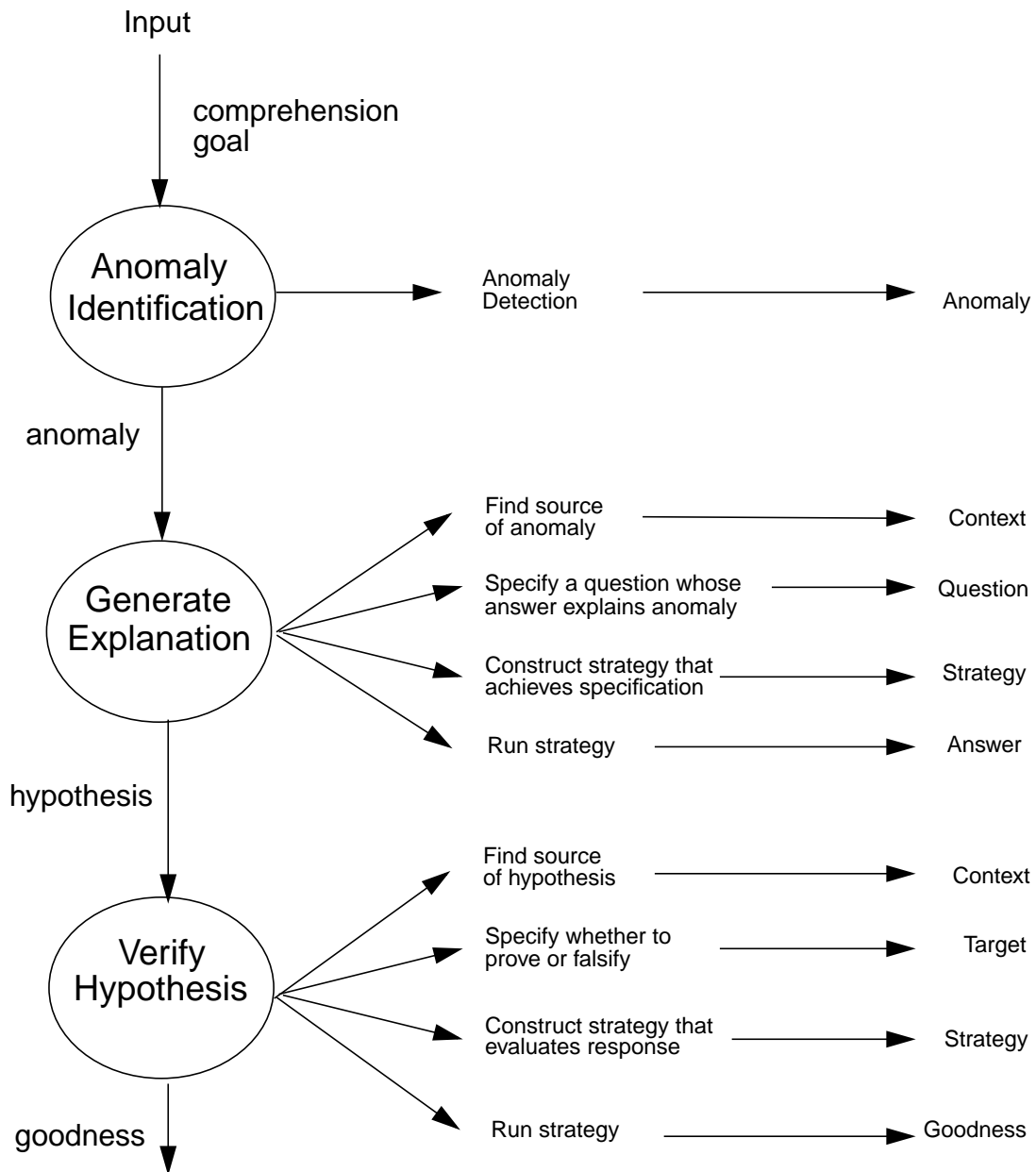


Figure 14. Question-driven understanding

reasoning in much the same manner as it processes input from the world. This enables a learner to explain its failures, decide what to learn, and then construct a learning strategy.

## **2.4 Prolog to Parts Two and Three: The content and the process**

Because of the peculiar relationship between content theories and process theories when explaining introspection, the material will necessarily be distributed somewhat throughout the following chapters, rather than occurring in strict, sequential order. Although most of the material concerning content theories and representations will come first, some of the representation of processes must await the chapters explaining the processes before full details can be presented. On the other hand, if the chapters on process preceded that of content, then some of the material would necessarily have to wait for the section on representation because some of the processes crucially depend on the structure of the representations. So where necessary, the following two parts of the thesis will provide explicit pointers to provide the interested reader with details concerning the relationship between the content and process theories. Forward references from Part Two (A CONTENT THEORY OF MENTAL REPRESENTATION) will point to details in Part Three (A PROCESS THEORY OF LEARNING AND INTROSPECTION), and backward references will provide the inverse function.



*Part Two*

***A CONTENT THEORY OF MENTAL REPRESENTATION***



## CHAPTER III

### SYMPTOMS AND CAUSES OF FAILURE: THE CONTENT

*The general idea of failure-based understanding is that examining how we make comparisons between our expectations and what actually occurs is the key to our knowledge of the understanding process itself.*

—Schank & Owens (1987), p. 203.

Failure provides both human and artificial reasoners with strong clues when deciding what needs to be learned (Birnbaum, Collins, Freed & Krulwich, 1990; Cox & Ram, 1994b; Fox & Leake, 1995a; Hammond, 1986; Hayes-Roth, 1983; Kolodner, 1987; Pazzani 1990b; Reason, 1992; Schank, 1982; Schank & Owens, 1987; Sussman, 1975; Stroulia, 1994; VanLehn, 1991b). One of the major goals of establishing a content theory of introspective learning, therefore, is to provide both a general characterization of reasoning failure and the potential causes of such failure in order to discover the nature of these clues. A sufficient characterization of failure will categorize the kinds of cognitively salient symptoms that signal to the reasoner that something worth learning exists. A sufficient taxonomy of the causes of failure will include those factors that account for each symptom in enough detail as to enable learning from them. The learner's task, then, is to perform an explanatory mapping from symptom to fault, and thus, to determine what causes a particular failure. Such explanations detail what needs to be learned by circumscribing the faults that must be corrected.<sup>17</sup>

---

17. This chapter's position does not claim that all learning is guided by failure. Success contributes to learning as well, but the impetus for learning resides entirely with failure in the theory of learning presented here. For alternative theories, see Siegler's evolutionary theory of learning in which the strategies, concepts, and rules that most successfully adapt under competition become associated to specific conditions (Siegler, 1991). See also Jones & VanLehn (1991) and VanLehn (1991b) for an additional counter-view, but see Appendix A, "THE DEGREES OF FREEDOM IN LEARNING" for a computational complexity argument for why failure may be preferred over success in learning.

Failure occurs in sundry ways: a seat belt malfunctions during an automobile crash on a rainy evening; a nuclear power plant experiences an unscheduled release of radioactive gasses; a student solves only 60% of the problems on a physics test correctly; a passage from Dostoyevsky is misinterpreted by a reader; and, like Andrew from the Walnut Cove cartoon (discussed in Chapter I, page 6), people hold incorrect expectations. But given these situations, with what perspective should these examples be best interpreted? That is, should the causes of failure be explained with reference to the external environment and contingencies that bear on the reasoner or with reference to internal factors of the reasoner?

For example, is the reason that a person is injured in a car crash because the seat belt fails or because the driver chose to drive too fast despite the rainy conditions? Do reactor gasses become injected into the environment because of mechanical malfunctions, because of poor design or because of operator failure? The position here is to focus upon failures made by the reasoner and the causes internal to the reasoner, rather than failures caused by external events and devices. Internal causes form the emphasis because this is the location over which the learner has personal control of the situation. It does no good to explain a failure in non-operational terms, if the goal is to improve performance (Owens, 1990b; Ram, 1989; Ram & Leake, 1991); instead, the reasoner must evaluate internal decisions and goals in order to change its mental world in the light of the situation and thereby to avoid repeating the failure indefinitely.<sup>18</sup>

All reasoning failures do not stem from incorrect reasoning, however. Often, it is a lack of attention or reasoning that contributes to mistakes. For instance, Andrew's failure originated in the lack of a mental event; he did not remember it was Sunday. Moreover, he was waiting for a bus that never arrived and so the external manifestation of the failure was unusual; it was the lack of an external event. Failure is not always calculating a wrong solution. Indeed, many of the wrong answers on a student's test may have been marked wrong simply because they were left blank. Thus, errors come in two varieties: errors of commission and errors of omission.

Although past research has developed domain-independent taxonomies of failure (e.g., Kass, 1986, 1990, specifies a taxonomy of failure during explanation and both Hammond, 1989, and Owens, 1990a, report a taxonomy of planning failures), much of this previous work is task dependent. To provide a content theory of introspective learning, this

---

18. Granted, this change in the internal environment may include new goals such as changing the configurations of the external world so as to make planning or reasoning more efficient. For instance, one may place paper filters in a location near the coffee machine in order to facilitate plans for the brewing of morning coffee. Although attention to such interactions will be minimized here, see Hammond (1990) for an approach to such task interactions and associated learning.



chapter provides a taxonomy of reasoning failure that is both domain independent and, to the greatest extent possible, task independent. After Section 3.1 presents a theoretical model of reasoning based on the generation of expectations, Section 3.2 will analyze the model to exhaustively enumerate the classes of failures implied by the model. Such classes of failure constitute the failure symptoms a reasoner should be able to detect. Given these classes of failures that systems or humans may perceive as symptoms, Section 3.3 taxonomizes the possible factors involved as causes of such classes of failure. The process theory that specifies how a system can map from symptom to fault will be deferred until Chapter VII (Section 6.2, “Blame Assignment: Explaining reasoning failure”). Instead, the current chapter provides a content theory for representing failure symptoms and causes (faults) used in the process theory, while the next chapter provides a formalism for representing this content in declarative structures.

### 3.1 A General Model of Expectation-Driven Reasoning

Given an intelligent system, reasoning is performed upon the representation of some input. Unlike the simple characterization depicted in Figure 15, the input is not just perceived, but in addition, an attention mechanism filters the input as determined by the current mental state of the reasoner. The crucial elements of the reasoner’s mental state are the goals and the expectations present in the reasoner’s memory. The filtered input, along with the reasoner’s knowledge, goals and expectations, then determine some interpretation of the input representation causing some additional goals and expectations. These conditions present a rich context from which to detect a failure.



Figure 15. A reasoner’s input

The reasoner is not just interpreting input, either. Rather than passively perceiving objects in the environment, an intelligent agent actively predicts future events surrounding such objects. Moreover, an agent deliberately performs actions in the world. Reasoning functions in support of efforts to understand the world and to achieve goals in the world. But, the world model the reasoner constructs is not confined to the narrow band of the present as determined by the current input; rather, the model spans the events in the immediate past as interpreted by experience and generates expectations of what the world will be

like in the immediate and far futures. Expectations enable the reasoner to be prepared for the future. The reasoner can thus avoid anticipated failures by generating contingency plans for them (Hammond, 1986, 1989). From the point of view of learning, however, the most interesting and valuable expectations are those that are violated, because these types of expectations provide the potential for improving the reasoner's anticipate-and-avoid behavior.

Thus, another fundamental purpose of forming expectations is to test the general limits of knowledge, independent of particular goals of the moment. That is, agents generate expectations to improve the boundaries of their knowledge: to retract those parts of the boundaries that are incorrectly extended and to expand the limits where gaps exist. An expectation represents a hypothesis or projection of current knowledge. When the hypothesis is falsified or when the projection is violated, the potential for self-improvement exists. One of the most basic mental functions, therefore, is to compare one's expectations with environmental feedback (or, alternatively, a "mental check" of conclusions). As a simple model of this comparison operation, consider Figure 16. The reasoner calculates some expected outcome and compares it with the actual outcome that constitutes the feedback.

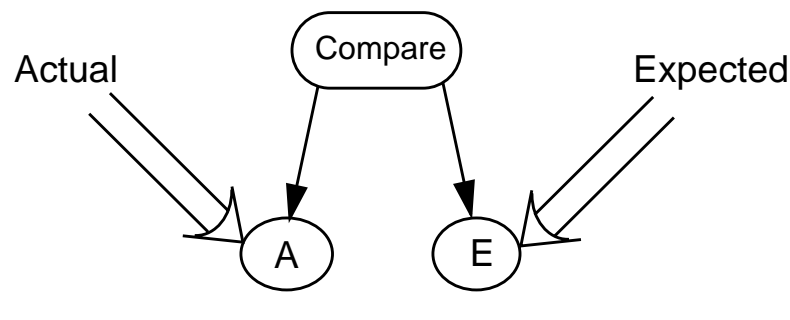


Figure 16. The basic comparison model

An *outcome* is defined broadly without reference to a specific task. The *expected outcome* could be the result of either a problem-solving process or a comprehension process. If it is a problem-solving process, the outcome could be in terms of a solution to a problem specification. For example, a problem may be specified as an operational goal to solve an eight-square puzzle. The solution is thus a series of transformations that end in the goal state. If the problem-solving task is a planning task, the outcome is a plan of actions that will accomplish a goal. If the problem is a design specification, the outcome is the proposed design that will satisfy the design function.

Comprehension processes, on the other hand, attempt to predict and understand events in a stream of input. Therefore, the outcome would be an interpretive understanding of a system's input, such as a reader's text comprehension of successive sentences or an art critic's visual comprehension of a painting. Both text and art can violate the observer's tacit expectations concerning what will be observed. In the drug-bust example from Section 2.1.1 for instance, the Meta-AQUA system implicitly expects dogs to bark at animate objects, even though it did not generate that expectation prior to encountering the sentence containing a dog that barked at luggage. To satisfy the comprehension task in complicated or unusual input, questions may be raised and an explanatory process may be invoked.<sup>19</sup> Hence, the expected outcome will be an explanation (Ram, 1991, 1994; Wilensky, 1983). That is, the reasoner consciously anticipates a certain explanation to be true of some object or event in the input which is to be explained. When these explanations prove incorrect, such as the explanation that the luggage threatened the dog, explicit expectations can be violated as well.

In addition, the specific mental process that forms an expectation (expected outcome) is not determined *a priori*. The process may be either an inferential process such as deduction, or it may be a memory process that retrieves an expectation from memory. For instance, to understand a story input, the reasoner (reader) may retrieve from memory a schema with which to interpret the story fragment.

Finally, the *actual outcome* may originate either internally or externally. That is, feedback may come from the environment (via perceptual/interpretive processes, of course), or it may emanate from a mental process such as an arithmetic check of a mathematics computation. In all such cases, the actual outcome is compared with the expected outcome in order to decide whether or not a failure exists in reasoning. If such a failure is detected, the reasoner attempts to explain the failure and to learn from it.

## 3.2 Types of Reasoning Failure

A *reasoning failure* is defined as an outcome other than what is expected (or a lack of some outcome or appropriate expectation). Such a definition, in light of the basic model above, presents a number of implications. Indeed, a logical matrix can be drawn depending on the values of the expected and actual outcomes. The expected outcome may or may not

---

19. The relationship between question asking and explanation is not always obvious. Sometimes the expected outcome is described as an explanation of an anomaly, while at other times the outcome is described as the answer to a question. The relationship in comprehension tasks is that anomalies cause questions to be posed of the form "Why did some event occur in the input?" The answer is an explanation that answers this question.

have been produced; thus, the expected outcome node,  $E$ , either exists or does not exist. Also, the actual outcome node,  $A$ , may exist or it may not. These values define a truth table as shown in Table 2.

Table 2: Logical truth table for reasoning model

	$\exists E$ expectation exists	$\nexists E$ expectation does not exist
$\exists A$ actual exists	Contradiction	Impasse
$\nexists A$ actual does not exist	False Expectation	Degenerate (N/A)

### 3.2.1 Four Basic Cases

Given this analysis, four basic conditions exist: contradiction, impasse, false expectation, and one degenerate case. The following four subsections will examine each in succession. Subsequent sections will introduce two more failure types (surprise and unexpected success).

Q6<sup>a</sup>: What kinds of reasoning failure exist?  
 Ans6: Contradiction, impasse, false expectation,  
 surprise, and unexpected success.

a. The numbering on the questions in boxes throughout the text refer to the subgoal structure in Figure 9 on page 14.

#### 3.2.1.1 Contradiction

If a system incorrectly understands some input, or solves some problem incorrectly, so that its expected interpretation or solution differs from the actual state of affairs (given some criteria or feedback), then a failure has occurred. This is a very conventional notion of failure and will be termed a *contradiction*. Contradictions are errors of commission since the reasoner generates an specific expectation that is subsequently proved false.

An obvious instance of a contradiction would be for a student to solve a physics problem incorrectly because of incorrect or missing assumptions. As a less obvious example,

another student may be told that the infinite series  $.999\overline{9}$  is equivalent to 1.0 (which is true). The assertion contradicts the student's naïve concept of numbers and the normal expectations arising with regular decimal series. Although this is not an overt error of commission as is the first example, it nonetheless represents the holding of an incorrect belief that contradicts a correct statement. Rather than questioning the statements of the teacher (i.e., the validity of the input), however, the good student will notice the inconsistency, pay closer attention to the lesson, and hopefully, question the validity of the student's own concepts and beliefs.

### 3.2.1.2 Impasse

An *impasse* is an error of omission defined as either a failure of a process (memory or inferential) to produce any outcome or as the condition under which no process is available to generate an outcome. If a reasoner is baffled when attempting to remember a fact or solve a problem, an impasse is said to have occurred. Andrew's episode of forgetting that it is Sunday is an example of a memory impasse. "Drawing a blank" on a brain-teaser is a problem-solving or inference example of impasse.

In the Soar model of general cognition, the impasse is a pivotal concept. Newell (1990) categorizes four types of impasses within the Soar architecture and enumerates them: tie impasse, no-change impasse, reject impasse, and conflict impasse. Events during Soar's decision cycle lead to each of these cases. For example, a tie impasse results when two or more actions are suggested by productions without preferences for one over the rest. A no-change impasse results when productions do not produce a significant change from the previous decision cycle. Each of these impasses will result in new subgoals to pursue, but are not explicitly considered failures. The taxonomy originates from an exhaustive analysis of the Soar decision cycle and are thus specific to the Soar architecture (or to similar decision control structures). Our impasse failure type is most similar to Soar's no-change impasse. We have not dealt with conflict resolution in any sophisticated way at the current time.

### 3.2.1.3 False expectation

*False expectations* occur when a reasoner expects an outcome that never occurs or is impossible. For example, a spectator may expect to see the launch of the space shuttle *Endeavor* while at Cape Canaveral, but engineers abort the launch. The spectator experiences a false expectation (perhaps even depression) when the launch time comes and goes with no takeoff. In the Walnut Cove cartoon, Andrew expects that the bus will come, but it does not. A novice theoretical computer scientist might expect that she has a solution to the "Halting Problem,"<sup>20</sup> not knowing that Turing proved many years ago that no such solution is possible.

### 3.2.1.4 The degenerate case

The cell marked as degenerate corresponds to the condition where a reasoner has formed a question or problem, has not generated an answer or solution, and has not been provided one by the environment. Despite the fact that a reasoner may consider this condition and dwell on the fact that no progress is being made in the reasoning, the case is not classified as a true failure. Instead, the problem or question is in either the state of active processing or of current suspension. Until either a solution or answer is generated or until one is provided by an external source, the failure cannot be said to have arisen.<sup>21</sup> Failure is detected in response to a comparison or after it is determined that a result is not really possible (i.e., false expectation). Therefore, this case will be considered degenerate in the matrix.

### 3.2.2 Extending the Analysis

The four cases above are sufficient to cover most of the model of Figure 16 (p. 44) if time is not considered in the equation. However, by considering that the reasoner's expected outcome and the actual outcome (the nodes E and A) may occur in two different orders, a new dimension emerges. A reasoner may determine an expected outcome in advance, or through hindsight given some feedback, may determine that one should have been produced previously. Consider Table 3.

Table 3: Expanded table for reasoning model

	$\exists E$ expectation exists	$\nexists E$ <b><math>\neg E</math> then A</b> expectation does not exist; feedback after knowing expectation does not exist	$\nexists E$ <b>A then <math>\neg E</math></b> expectation does not exist; feedback before knowing expectation does not exist
$\exists A$ actual exists	Contradiction	Impasse	Surprise
$\nexists A$ actual does not exist	False Expectation	Degenerate (N/A)	Degenerate (N/A)

20. The problem is to predict whether an arbitrary program will successfully return control or whether it will enter into an infinite loop.

21. It is arguable that if a reasoner is reminded of a long outstanding problem, in effect, it could be called a failure. Moreover, if time constraints are included in the analysis, then this case may also be considered a failure. For example, if a solution is not produced by a student during an exam within a certain time increment, then it is marked wrong and no feedback is presented by the evaluator.

### 3.2.2.1 Surprise

When a system has no explicit expectation, yet an event occurs which should have been expected, then a *surprise* exists. In these cases, the reasoner considers the node, A, before considering the existence of E; that is, before the reasoner realizes that the expectation, E, should have been generated. Using hindsight, it is up to the reasoner to recognize that there should have been a problem to solve or that the reasoner should have tried to remember to do something.

For example, a robot may never infer that it needs to refuel its vehicle despite the fact that it has viewed the gas gauge steadily drop (example borrowed from Owens, 1991). When it eventually runs out of gasoline, it is able to deduce that the incident stems from earlier failure to formulate the refueling problem and subsequently solve it by planning to obtain the resource.

### 3.2.2.2 Unexpected success

Finally, if one considers that the comparison operation actually produces a value, then this analysis produces another failure type. The value of the comparison is some relation between the expected outcome, E, and the actual outcome, A. In Figure 17, this relation is marked as the node R. When reasoning is successful, then this value should of course be equality; that is, the relation, R, in Figure 17 should be the “=” value. When reasoning is unsuccessful, this value will be the “≠” value.

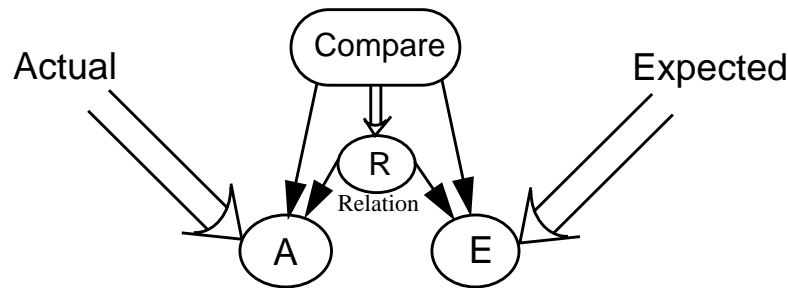


Figure 17. The extended comparison model

Reasoners are able to make predictions about this value (i.e., predict whether they will be successful at a future reasoning task). The normal condition is that all reasoners expect to succeed in their reasoning goals, but this is not always the case. So the expectation may actually be a prediction of that value. To be specific, the reasoner expects the value of E to be something other than A, the actual outcome. Thus, they expect R to be the “≠” value.

This condition expands the table a final time, producing Table 4.

Table 4: Final table for reasoning model

	$\exists E$ expectation exists	$E = \neg A$ $R = \text{"}\neq\text{"}$ expectation is opposite of actual; i.e., expects comparison is not-equal relation	$\nexists E$ $\neg E$ then $A$ expectation does not exist; feedback after knowing expectation does not exist	$\nexists E$ $A$ then $\neg E$ expectation does not exist; feedback before knowing expectation does not exist
$\exists A$ actual exists	Contradiction	Unexpected Success	Impasse	Surprise
$\nexists A$ actual does not exist	False Expectation	Degenerate (N/A)	Degenerate (N/A)	Degenerate (N/A)

If a reasoner expects that it will not be able to compute any answer or the correct answer, but it does nonetheless, then another failure class exists called an *unexpected success*. Reasoner are obviously aware of their comparisons if performed with deliberation, so the reasoner may actually anticipate the accuracy of their predictions. Therefore, if a reasoner expects that the comparison will show inequality, then the reasoner is anticipating that performance will not be successful. That is, the reasoner expects to fail in a future reasoning task, yet succeeds nonetheless.

Although one normally would consider this type of a failure to occur during problem solving or planning, it may also happen during memory performance. Metamemory studies show that humans can predict whether or not they will remember items well. See, for example, the experimental studies of feelings-of-knowing, i.e., judgements of future recognition of an item that was not recalled during some memory test (e.g., Krinsky & Nelson, 1985) and judgements-of-learning, i.e, judgements at rehearsal time as to future memory performance (e.g., Nelson & Dunlosky, 1991). Therefore, an agent may predict that memory will fail on a given item when, in practice, retrieval succeeds. Like the representation of contradiction, the agent expects one outcome (failure), yet another occurs (success) during an unexpected successes.

### 3.3 Causal Factors in Reasoning Failure

The purpose of the preceding material is to enumerate the kinds of failure a system should be able to anticipate during reasoning. These types of failures (contradiction, impasse, false expectation, surprise, and unexpected success) constitute the salient features



during reasoning that an intelligent should be able to detect and then explain. Here we enumerate the types of causes from which a system has to choose when constructing such explanations of reasoning failure. The task of explaining failure is to map members of one taxonomy (failure symptoms) to members of another (failure faults or causes).

To organize the possible sources of reasoning failure, it is again necessary to consider reasoning in terms of general assumptions. Clearly, reasoning is intentional and thus, oriented toward the pursuit of specific goal states. Moreover, we assume that reasoning uses knowledge to process perceived input from the environment in order to create a representational state of the world and to achieve these desired goals. Reasoning processes transform specific mental states into new mental states. Some of these states are knowledge states representing facts and experience, some are perceived states representing conditions in the external environment, and some are goal states representing desired new states in the environment. Based upon such representations, reasoning produces decisions that result in actions that change the world, thus producing new environmental states that can subsequently be input or perceived by the reasoner in order to compare the goal to the actual state of affairs in the world. Such decisions result in new internal actions that may change the expectations present in working memory that bias later input. Given these assumptions, reasoning will fail if any of the constituents of reasoning fail; that is, if a problem exists with the reasoner's *knowledge*, *goals*, *mental processes*, or input from the *environment*.

### 3.3.1 Selection Factors

In addition, not only can these components be a likely cause of error, but the ways in which the reasoner selects them can also be a source of error. Non-selection is an important and often overlooked factor in the analysis of failure. It is a result of poor memory organization rather than incorrect memory content. That is, failure can occur, not because an agent does not know some fact, but because the agent cannot retrieve the fact when needed.

Computer memory is sometimes viewed as a virtually error-free medium in which retrieval of data is performed by simple fetch operations. As computer memories grow, however, brute-force search for the address to perform the fetch becomes increasingly intractable. Memory indexing is added in order to make memory retrieval more efficient. A memory-indexing mechanism is a trade-off between time to search and accuracy of retrieval; although efficiency is gained, indexing schemes risk not finding the proper information. That is, given some query, a computer may not find or be able to select a knowledge item, a suspended goal, or a reasoning strategy at all — from the user's point of view, it can "forget."

The *indexing problem* (Domeshek, 1992; Kolodner, 1984, 1993; Owens, 1993; Schank, 1982; Schank & Osgood, 1990) is that of choosing cues, or features in an input, to

be used as indexes for retrieving from memory the knowledge structures necessary to process an input. The converse problem, is the *problem of forgetting* (Cox & Ram, 1992a). If the cues are not chosen with care during retrieval time, or if the indexes are not chosen well during encoding, the reasoner may not recall a memory structure when it is needed. The forgetting problem is to reorganize memory and the indexes by which memory is accessed. Because reasoning failures may occur due to faulty memory organization, as well as because of faulty reasoning components or faulty knowledge, the selection or retrieval of knowledge plays an important role in the determining of cause of failure.

### 3.3.2 A Taxonomy of Reasoning Failure Causes

Table 5, “Detailed taxonomy of causes of reasoning failure,” presents a matrix for relating the causal factors that bear on the determination of blame. As indicated at the heading in the uppermost row, the table is divided into four major causal categories. Failure could stem from the knowledge states with which the reasoner makes decisions, goal states generated during reasoning, the reasoning processes used to achieve the goals, or the input that represents the environment and from which feedback is provided. In each of these categories, the relevant item may be either missing or wrong.<sup>22</sup> That is, omission errors occur when a necessary component is not present (this is represented by the “Absent” row in the table); whereas, commission errors occur when an incorrect component is present (this is represented by the “Wrong” row in the table). In addition, because knowledge is imbedded in a memory and must be retrieved before it can be used to pursue a goal, an error of omission can result from non-selection, rather than simply nonexistence. If an item is correct, then that category contributes nothing to the failure.<sup>23</sup>

For each dimension represented by a column in the table, a general characterization of it also exists in the last table row. These will be explained as entries in the table are discussed. The subtable suggests that similar causal factors are attributable to perceived agents (see Section 3.3.6). Although Table 5 organizes many factors in a coherent fashion, the task of identifying which of the factors are responsible for blame is clearly a complex one, especially when multiple causes exist.

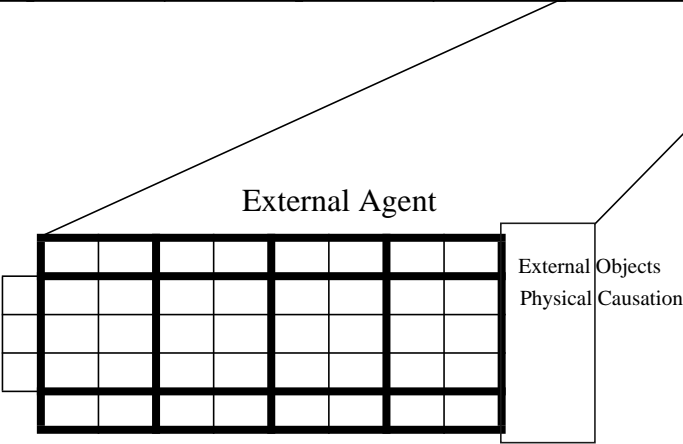
---

22. Note that to characterize a component as *incomplete* is actually a diagnosis of a component missing at some finer level of detail. Incomplete is therefore not a row of this table.

23. One of the targets of this research has been to produce representations for the cells of Table 5. Explicit Meta-XP representations are provided in Section 4.4, “Meta-Explanation Patterns,” starting on page 79. The representations of the various cells are chained into composite structures that capture typical failure patterns that occur during reasoning.

Table 5: Detailed taxonomy of causes of reasoning failure

	Knowledge States		Goal States		Processes		Environment		
	Domain Knowledge	Knowledge Selection	Goal Generation	Goal Selection	Processing Strategy	Strategy Selection	Input	Input Selection	
	Absent	Novel Situation	Missing Association	Missing Goal	Forgotten Goal	Missing Behavior	Missing Heuristic	Missing Input	Missing Context
		Wrong	Incorrect Domain Knowledge	Erroneous Association	Poor Goal	Poor Selection	Flawed Behavior	Flawed Heuristic	Noise
	Right	Correct Knowledge	Correct Association	Correct Goal	Correct Association	Correct Behavior	Correct Choice	Correct Input	Correct Context
		Theory	Memory	Desires	Opportunity	Action	Control	Perception.	Attention



For three of the columns: domain knowledge; goal generation; and processing strategy, a natural dualism is present in their interpretation. For example, strategies represent both mental and physical actions. Thus, there exist mental actions such as operators for mental arithmetic (e.g., integrate by parts in the calculus domain of the LEX program; see Mitchell, Utgoff, & Banerji, 1983) as well as physical actions like robot navigation schemas (e.g., avoid-static-obstacle; see Arkin, 1987). For each type of action, associated heuristics are used by a reasoner to choose when to apply the action. In a similar fashion, there is a physical and mental manifestation of goals and domain theories. Thus, an agent can have mental reasoning goals, such as “remember where I parked the car,” and can also have goals to achieve states in the world, like “be at my car’s location.”<sup>24</sup> Likewise an intelligent agent can have knowledge about the world as well as self-knowledge. Although these nuances are important distinctions to observe, as discussed earlier, the primary treatment presented here will concentrate on internal mental factors rather than external causes.

Despite the focus on the deliberative and top-down components of thought, rather than the data-driven or situation-specific factors, we cannot deny that bottom-up factors affect both reasoning and learning. As a research strategy, however, external factors will be minimized or ignored to provide scope and focus. Such a position is consistent with traditional cognitive science perspectives (see arguments in support of this position by Gardner, 1987; Hayes, Ford & Agnew, 1994; and Newell & Simon, 1972), although the emphasis is indeed at odds with some recent stances, such as the situated cognition paradigm (e.g., Clancey, 1991; Suchman, 1987).

Q4: What can cause reasoning failure?  
 Ans4: Knowledge, goals, processes, input, and  
 the way each are selected.

The remaining material of Section 3.3 will examine each of the four major categories in some depth, and provide examples of how failure can result from them. The first section on knowledge states is the longest, but much of the discussion will also apply to aspects of the remaining sections. The final subsection will also provide constraints upon the causal factors that assist in making the task of blame assignment more tractable. The chapter will close with a brief summary.

---

24. Note that the former is a subgoal of the latter.

### 3.3.3 Knowledge States

The domain knowledge of a system represents its theory of the objects, relations, and actions in the domain. This theory consists of facts and propositions in some declarative representation regarding the entities in the domain (i.e., its semantic knowledge of the domain) and some record of the events that have occurred during its experience with the domain (i.e., its episodic knowledge of the domain). In addition, since knowledge is maintained in memory, retrieval or selection of knowledge can also contribute to failure.

#### 3.3.3.1 Domain knowledge

The most basic type of failure occurs when the system's domain knowledge is at fault. A domain theory presents the rules, concepts, and relations involved in a particular self-contained knowledge system. For example, the classic cup-domain (Mitchell, Keller, & Kedar-Cabelli, 1986; Winston, Binford, Katz, & Lowry, 1983) provides inference rules used to identify household objects in the cup category (e.g.,  $\text{cup} \Leftarrow \text{stable} \wedge \text{liftable} \wedge \text{open-vessel}$ ;  $\text{stable} \Leftarrow \text{has-bottom} \wedge \text{flat-bottom}$ ).

A domain theory is considered incomplete<sup>25</sup> if pieces of the knowledge base are missing. Therefore, a *novel situation* represents an error of omission such that a reasoner has no knowledge with which to interpret some input or with which to solve some problem. In rule-based systems incompleteness occurs when a rule or an antecedent of a rule is missing, while frame-based systems are incomplete when concepts or attributes of concepts are missing.

Alternatively, a domain theory is considered incorrect if there are pieces of the knowledge base present that should not be. *Incorrect domain knowledge* is an error of commission occurring when incorrect knowledge erroneously biases an interpretation of some input or when incorrect knowledge is used to make faulty inferences in problem-solving. In rule-based systems this occurs when an extra rule or antecedent of a rule is present, while frame-based systems are inconsistent when concepts or attributes are present that should not be.

Rule-based domain theories are *overly specific* when they are missing some rules or when they possess extra antecedents (in a frame system, this entails missing concept types

---

25. Note that the use of incompleteness as a logical term is different. A logically incomplete domain theory is one in which a positive example of a category cannot be proven. This may occur if an extra antecedent on a rule exists, not just when rules are missing. A missing antecedent does not itself lead to logically incomplete theories.

or extra preconditions or constraints). Such theories are overly specific because either the inclusion of a missing rule or the deletion of an extra antecedent would include previously rejected positive examples. Domain theories are *overly general* when they are missing antecedents or when they possess extra rules (in a frame system, this entails missing preconditions or extra types). Such theories are overly general because either the deletion of an extra rule or the inclusion of a missing antecedent would reject an example that is mistakenly included in the concept (Mooney & Ourston, 1994).

In the drug-bust scenario (Section 2.1), Meta-AQUA's *dog-bark* concept is overly specific because it possess a constraint that the object of the action is animate. In effect, it states that dogs bark only at animate objects. By the classification of Table 5, this condition is termed incorrect domain knowledge.

```
(define-frame dog-bark
  (isa      (value      (event)))
  (actor    (default    (dog)))
  (object   (constraint (animate-object)))
  ...)
```

As a result, the input token (a dog barking at a piece of luggage) is rejected as a member of the *dog-bark* type and considered anomalous. Actually the system should have accepted the input example as a member of the *dog-barks* concept; however, the concept was overly specific because of the constraint on the object slot. A contradiction was the result. If Meta-AQUA used a rule-based domain theory, then this error would be caused by an extra antecedent such as in the following Horn-clause rule.

$$\text{dog-barks} \Leftarrow \text{actor}=\text{dog} \wedge \text{object}=\text{animate} \quad (1)$$

The important point to consider here is that the error type is implementation independent. The taxonomy of causal factors in reasoning failure depends on neither a frame-based nor a rule-based representation.

### 3.3.3.2 Knowledge selection

Because the knowledge of any non-trivial domain is extensive, exhaustive search is usually computationally prohibitive. In response, many knowledge bases are organized by associative indexes that link particular cues in the contexts with relevant knowledge. A failure may therefore occur, not because relevant knowledge does not exist for the reasoner, but

rather, because the knowledge cannot be selected from memory due to poor indexing. Thus, a *missing association* will cause an error of omission (the correct memory element will not be retrieved), whereas an *erroneous association* will cause an error of commission (the wrong memory element will be retrieved).

Note that every erroneous association necessarily implies a missing association; that is, if a system retrieves an incorrect memory item, it must be the case that it did not retrieve the correct item. If the proper index had existed, then the correct item would have been retrieved. Therefore, the correct index must have been missing.<sup>26</sup>

Like domain theories, indexes can also be overly general and overly specific. Consider that two ways exist in which one can think of indexing errors. Say the following attribute values exist: Red, blue, green, black, white, brown, gray, yellow, orange, and purple (see Figure 18). Assume also that red, blue and green are the primary colors; black and white are the shading colors; and the rest are the irregular colors.

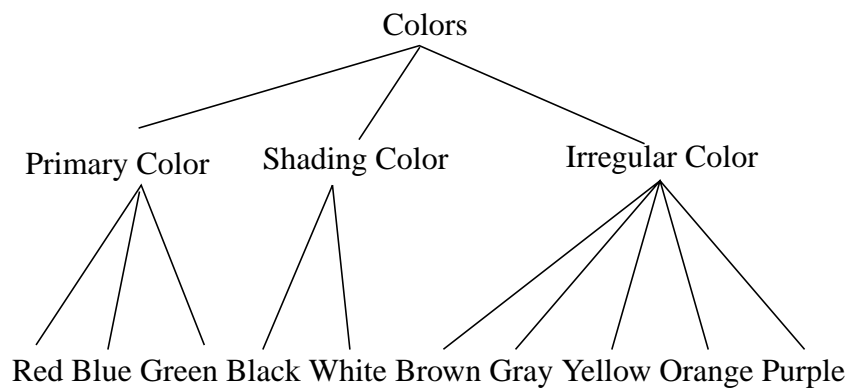


Figure 18. Example generalization hierarchy for color

---

26. The possibility exists that two indexes were matched, one pointing to the correct information and the other pointing to the erroneous information. Given some conflict resolution, the incorrect one is then chosen. In this case, one is tempted to say that no missing index is implied. However, despite the fact that IML theory does not directly address conflict resolution (nor does Meta-AQUA implement it), the index considered missing is one whose antecedents or preconditions would have matched accurately enough to avoid the resolution process.

Now a memory item, *M*, indexed by the shading colors could either be of the two representational forms below.

Black  $\rightarrow$  *M* (2)

White  $\rightarrow$  *M* (3)

That is, either the above two indexes (2 and 3) can represent the association, or the index below (4) can.

Shading\_Color  $\rightarrow$  *M*. (4)

Index 4 is a generalized index and equivalent to the previous two simple indexes. In addition, suppose that an indexed explanation exists that asserts red and blue composes to the color purple. This explanation, *E*, might then be indexed by the following three indexes:

Red  $\rightarrow$  *E* (5)

Blue  $\rightarrow$  *E* (6)

Purple  $\rightarrow$  *E* (7)

Retrieval failures may now occur in a number of ways. If, instead of Index 5 and Index 6, the index *Primary\_Color*  $\rightarrow$  *E* exists in their place, then this index is overly general. The explanation, *E*, will be brought to bear at inappropriate times. Such a condition is equivalent to the erroneous association, *Green*  $\rightarrow$  *E*, being added to the existing indexes. On the other hand, if Index 2 is missing, then Index 3 must be overly specific when retrieving the memory item *M* (i.e., Index 3 should really be Index 4). Therefore, a missing association is equivalent to an overly specific index because it does not match the cues in the context at retrieval time.<sup>27</sup>

As another illustration, consider the second example in the drug-bust scenario from

---

27. Note the family resemblance between these two knowledge selection failures, erroneous association and missing association, to the domain knowledge failures of incorrect domain knowledge and novel situation respectively. This resemblance permeates across the columns of Table 5.



Chapter II (Section 2.1.2). In this episode Meta-AQUA forgets the earlier constructed explanation that the dog barks when detecting contraband; that is, an impasse is reached when no explanation come to mind. The reason for the impasse is that the explanation had earlier been indexed by dogs barking at containers, which is overly specific when in the new context of a dog barking at a laundry pile.

### 3.3.4 Goal States

Goals are distinguished knowledge states that focus the reasoning and provide the reasoner with a specific target state to achieve. They represent the desires or intentional states held by a system. If a reasoner fails to generate an appropriate goal or subgoal, or if the reasoner generates an inappropriate (sub)goal, then reasoning will not likely succeed. System resources will be expended on tasks not likely to provide progress in the profitable directions. In addition, if the reasoner cannot immediately achieve a goal, the goal may be suspended and indexed in memory with the hope that an opportunity may arise with which to achieve the goal in the future. As is the case with normal knowledge states, such suspended goals are subject to retrieval failure, if they are not indexed in such a way as to correspond to the contexts within which opportunities await.

One may object to this taxonomic category on the grounds that a goal is just another knowledge state and not any different when explaining failure. For example, a fact may exist in the knowledge base that a particular terrorist was convicted. In addition, a prosecuting attorney may have the goal to achieve the state of another terrorist being convicted. One might argue that very little difference exists in the structure of the two pieces of knowledge. However, major differences exist, not just with the semantics of each, but with the actions a system performs in response to each. When a system determines the cause of its failure is that it had an incorrect goal, then it must continue the blame assignment to determine why it posted such a goal; whereas, if the system determines that the reason it fails is due to some specific knowledge, then this is often sufficient for the blame assignment task.<sup>28</sup>

#### 3.3.4.1 Goal generation

In goal-driven intelligent systems, two obvious causes of failure are that the correct goal was not generated and that the wrong goal was generated (Owens, 1990b, 1991). If a problem exists in an agent's world, then the reasoner should detect it and generate a goal to

---

28. It is important to avoid arbitrarily long chains of meaningless causation. One heuristic is actually more specific than indicated above. A system can stop blame assignment if encountering a knowledge element in the BK; the process continues if the item is an element of the FK or is a goal. More details are provided in Section 6.2.

solve it. But, if the problem is never detected, the goal will not be generated. This cause of failure is called a *missing goal*. If, on the other hand, a goal is generated to solve a problem that is not actually a problem, then the cause of failing to solve it is called a *poor goal*.<sup>29</sup>

Goals are also generated during a comprehension task to more fully understand those inputs which are anomalous or otherwise interesting. Such understanding goals seek to explain the particular input state by creating causally connected links to prior states and events in the input stream. Therefore, a missing goal can be a failure to detect an anomalous input state and a poor goal can be incorrectly characterizing an input state as anomalous when not. Subsequent goals may thus be misguided.

Often we need to look for why the goal was not spawned or why the wrong one was spawned by looking at the knowledge, so a relation between these categories exist. Note that goals are members of the FK (except when suspended), whereas knowledge is stored in the BK.

### 3.3.4.2 Goal selection

Unfortunately, even the most appropriate goals cannot always be pursued at the moment they are generated. Sometimes the resources or knowledge necessary for goal pursuit are not available to the reasoner, and so the reasoner must wait until the resources become available. Rather than wait indefinitely, an opportunistic reasoner (Birnbaum & Collins, 1984; Hammond, 1988; Hammond, Converse, Marks & Seifert, 1993; Hayes-Roth & Hayes-Roth, 1979; Ram, 1989; Simina & Kolodner, 1995) will suspend blocked goals, store them in memory, and pursue other goals until the time the resources are present.<sup>30</sup> Therefore, as with any item stored in memory, suspended goals may not be retrieved or selected at the moment they are needed by the reasoner. Goals, like domain knowledge, are subject to the problem of forgetting when the indexes created at storage time do not match the cues present in the context at retrieval time.

---

29. Note that the wrong goal may be generated, while at the same time an incorrect goal may be missing. However, unlike knowledge selection errors, this is not necessarily the case. The errors are independent.

30. More specifically, goals that are spawned during reasoning exist in the FK. If the goals cannot be achieved immediately, however, they may be transferred to the BK, along with a trace of the reasoning that spawned them, in order to suspend the processing until their achievement is more likely (e.g., when the preconditions upon which they depend become available). During such this process, the goals are indexed in the BK by features selected to match elements of the environment that are characteristic of conditions likely to exist when resumption of the goal pursuit is profitable. When a goal is resumed, it is returned to the FK from the BK.

A *forgotten goal* is a suspended goal that was not retrieved from memory at the appropriate time (this cause is equivalent to a missing association). Alternatively, a failure is caused by a *poor selection* when the goal selected from memory is inappropriate given the current context (this cause is equivalent to an erroneous association). As with the knowledge selection category, if an inappropriate goal is selected, then it must be the case that the appropriate goal was not selected (i.e., poor selection implies forgotten goal).

As an example of a forgotten goal, a member of an audience listening to a lecture may want to ask a question of the speaker. Instead of interrupting the speaker, the listener may decide to wait until the end of the lecture. After all, the speaker may answer the question in the remainder of the presentation. However, at the end of the presentation, the listener may not remember to ask the question, if it was not already answered. The cues available at the end of the lecture may not be sufficient for retrieving the question from memory.<sup>31</sup>

### 3.3.5 Processes

The process column considers those factors in reasoning which produce mental actions and transformations of knowledge. That is, it considers the soundness of the reasoning itself. The choice of which reasoning strategy to use in a given situation is a matter of control of mental action. In many AI systems (e.g., PRODIGY, Minton, 1988), the control is by means of some heuristic rules or triggering mechanisms.

With respect to humans, Reder (1987) has shown that after reading a story, human subjects use different strategies to answer questions posed about the story and vary the strategies used depending on existing conditions such as the time from reading the story to asking the question. In some cases, a person will use a direct memory retrieval method while at other times a person will infer a plausible answer based upon related facts. Moreover, her report also supports the existence of a specific strategy-selection stage of question answering that automatically evaluates the knowledge relevant to the question and then deliberately decides on a strategy. Siegler (1988) has reported similar strategy choices with respect to addition, subtraction, and reading skills in children.

#### 3.3.5.1 Processing strategy

Candidate reasoning processes can be either fine-grained (e.g., a problem-solving operator) or large-grained (e.g., a reasoning paradigm such as means-ends analysis). A relation between domain knowledge and the processing strategy column is that an error

---

31. Note that variants exist when forgetting a goal. Forgetting to do some action is equivalent to what psychologists refer to as failure of prospective memory (Reason, 1992).

could be a missing precondition in an operator rather than a missing operator itself.

If a process is incorrect by the inclusion of some faulty rule, rule antecedent, assertion, precondition, or incorrect sequencing of operators, then the error of commission is termed a *flawed behavior*. If, on the other hand, a reasoning process is missing a rule, rule antecedent, assertion, or precondition, then the failure is called a *missing behavior*. For example, a failure may be due to an agent lacking a specific skill to achieve a goal.

### 3.3.5.2 Strategy selection

For processing strategies the organization is captured by heuristic rules that link applicability conditions with some operator or process. Reasoning strategies are applied only if they are selected using some heuristic that determines they are applicable. Thus, the heuristics can be thought of as “indexes.” So, as with the other selection columns considered so far, a failure may occur, not because the reasoner does not have the strategy with which to process the input, but rather because it does not have the specific heuristic to signal the strategy’s applicability, or because another heuristic selects a competitive strategy.

A failure that happens due to non-selection of a given process is an error of commission called a *missing heuristic*. Alternatively, if a process is chosen by mistake, then the *flawed heuristic* error of commission is said to have occurred. As with previous selection failures, if the wrong process has been chosen, then it must be the case that the correct process has not been selected; that is, a flawed heuristic implies a missing heuristic.

For example, the kinds of explanations that Meta-AQUA may produce are determined by the methods of explanation as well as the knowledge in its BK. During the reading of the drug-bust story, if Meta-AQUA had decided to use case-based reasoning rather than XP application, the explanation of why the dog barked at the luggage may have been sufficiently different.

### 3.3.6 Environment

The most complex column is the one representing the input to the system from its environment. The major distinctions in this dimension are between perception (or what is input from the environment) and attention (what is selected for further processing by the reasoner from the perceived input).

The input to a system constitutes the interface between the internal world (cognition) and the external world (environment). As suggested by the sub-table, if one allows interaction with other agents in the world, then there may be blame associated with the goals, strategies, input and knowledge of other agents. Thus, for example, noise in the input may

actually be due to deception by opponents caused by conflicting goals of the external agent. This makes blame assignment exceedingly difficult. However if one is to categorize an open world, this source of blame must be acknowledged. The input column is most interesting if one considers that there exists an input, the perception of the input, and the interpretation of the perception of the input. However, for the purposes of brevity and focus, the treatment in the following sections will often ignore such distinctions, concentrating on the analysis of the main table alone.

### 3.3.6.1 Input

*Noise* is considered a fault if a piece of the input from the environment is incorrect. It is perhaps unusual to consider this failure as an error of commission since the reasoner may not have caused the noise. However, it is useful to think of the error as being committed by the environment itself (e.g., faulty data-collection equipment). *Missing input* is an error of omission by the environment such that some critical piece of information is not present in the perceiver's environmental context.

As an example of noise, an agent may be told that a deceased individual came back to life (which is false). Because the input contradicts the firm belief in people's mortality, the input should be questioned. In this case the agent may question the validity of the statement because it contradicts a firm expectation. However, unlike faulty equipment that may cause noise in an input, this example is intentional.<sup>32</sup>

### 3.3.6.2 Input selection

Unlike the errors discussed in the immediately preceding subsection, the errors of input selection are errors attributable to the reasoner, not the environment itself. The lack of attention to proper element in the input stream can cause reasoning to go astray. A error of omission is therefore called a *missing context*; whereas, an error of commission is *incorrect context*. Again, like previous selection columns, an incorrect context implies a missing context.

Unlike the other selection columns, however, the causal factors represented by input selection are not memory organization problems; rather, they are attentional mechanisms (possibly related to goals and expectations). An error may occur, not because there is not a requisite piece of information in the perceptual field (i.e., input) of an agent, but rather, because the agent does not focus on the information or consider it properly.

---

32. In general, see Johnson & Seifert (in press) for recent research regarding the effects of misinformation on human inferences and judgements.

In summary, failures can be caused by either knowledge states, goal states, processes, or by environmental input. In each case, a distinction exists between the component being at fault or the selection of the component being at fault. Furthermore, in each of these eight categories, the item may be either wrong or missing. The result is sixteen causes of failure that may explain reasoning error, plus combinations of each.

### 3.3.7 Causal Invariants

Although the table of reasoning failure causes (Table 5 on page 53) is complex, the number of possible explanatory combinations generated from the table is not the full permutation of the number of cells in the table. Some conjunctions are not possible because a number of invariants and constraints exist within the table. Five are listed below.

1. It is not possible to have an error of commission without also having an error of omission in all of the selection categories. This has been discussed in the previous subsections on selection.
2. It is not possible to have both a correct association and an erroneous association. If the correct cell is enabled, then neither of the other two cells in the column can coexist.
3. If both factors within a given column are present (missing and wrong), each necessarily corresponds to separate objects. A single knowledge token cannot both be correct and wrong.
4. It is not possible for an erroneous association to exist without either a novel situation or a missing association also present. The reason is that an erroneous association signals an expectation failure: something was retrieved that should not have been. Something else should have been retrieved instead. Thus, something is either not present in the domain knowledge (and therefore cannot be retrieved) or it was present, but no association was present with which to retrieve it.
5. A novel situation trivially implies a missing association. If there is no item in memory, then there cannot be an index to organize it in memory. The same can be said for missing goal (implies forgotten goal), for missing behavior (implies missing heuristic) and for missing input (implies missing context).

These constraints on the combinations of causal values assist a system when attempting to map from failure symptom to explanatory fault.

### 3.4 Summary and Discussion

This chapter described a general model of expectation-driven reasoning, extracted from the model a set of failure types or symptoms, and then constructed a taxonomy of causal factors that can explain individual reasoning failures from this derived set. Given this model of reasoning, we claimed that a learner must be sensitive to the symptoms of contradiction, impasse, false expectation, surprise, and unexpected success. When explaining such failure symptoms, a learner must be sensitive to the knowledge it used, the goals it pursued, the process with which it reasoned, and the environment within which the reasoning was oriented. Such taxonomies of failure symptoms and faults do not originate from an *ad hoc* compilation of everyday mistakes with a *post hoc* generalization over this list (i.e., a kind of “list and induce” method); rather, they stem from a reasoned analysis of ideal models of intelligent performance. This method of developing content-theory taxonomies represents an improvement over previous intuitive methods of analysis. One can challenge the taxonomy (or compare it with others) by examining either the assumptions or the analyses of the model. Without such arguments, a comparison of alternate lists of examples or categories is meaningless.

Along the way, individual sections presented a number of examples from real-life, from artificial domains, and from the Meta-AQUA implementation to argue in support of these basic results. The reasoning model is specific to neither domain nor task and contains no unreasonable assumptions. Therefore, the set of failure types derived from the model represents a testable hypothesis from which experimental methods may determine the reasonableness of the model. Although this dissertation will not attempt to test this hypothesis with human subjects, the following chapter will put forward functional justifications for why these categories are useful in computational systems. Chapter IX will support the hypothesis with empirical results from computational studies rather than psychological methods.<sup>33</sup> See Chapter X, FUTURE RESEARCH (Section 10.1.7, “Failure Types as a Cognitive Category”), however, for speculation as to a possible inquiries that may be made in conjunction with experimental psychologists.

This chapter has provided the theoretical constructs that support a theory of introspective multistrategy learning. Because the theory chiefly concerns learning, the concentration of analysis has been on how performance systems can fail and what constitutes the causes of such failure. Only through an analysis of such features can a system be developed which can learn to avoid such failures. This chapter has therefore described the content of the the-

---

33. Chapter IX will also entertain two additional hypotheses and describe experiments with the Meta-AQUA system to test these hypotheses. The hypotheses are that (1) introspection facilitates learning and (2) IML theory represents a sufficient explanation for human introspection.

ory from which the following chapter will give a concrete vocabulary and representation.



## CHAPTER IV

### MENTAL STATES AND MECHANISMS: THE REPRESENTATION

*But the number of those which are simple and primitive is not very large. For, in making a review of all those in which I have enumerated, we may easily notice that there are but six which are such, i.e. wonder, love, hatred, desire, joy and sadness; and that all the others are composed of some of these six, or are species of them. That is why, in order that their multitude may not embarrass my readers, I shall here treat the six primitive passions separately; and afterwards I shall show in what way all the others derive from them their origin.*

—Rene Descartes, (1649/1955), p. 362.

An early tenet of artificial intelligence is that reasoning about the world is facilitated by declarative knowledge structures representing salient aspects of the world. A declaratively represented world is easier for an intelligent system to understand and operate within than one in which knowledge is encoded procedurally or implicitly. The system may inspect and manipulate such structures, the system can be more easily modified and maintained (by either its programmer or itself), and such representations provide computational uniformity.<sup>34</sup> Furthermore, if a system is to reason about itself, the above tenet can be applied to representations of its own reasoning and knowledge. The aim of this chapter, therefore, is to begin to outline a declarative representation of mental activity. The goal is to explicitly represent the mental world that reasons about the physical world, just as past research has explicitly represented the physical world itself. Instead of representing states and events in the physical world, this chapter discusses how and at what grain level one should represent mental states and mental events. Given such representations, learning pro-

---

34. Proponents of procedural representations have argued against these points and countered with advantages of their own. See Winograd (1975/1985), especially his argument that second-order knowledge is easier to represent procedurally. See Stein & Barnden (1995) for arguments in favor of the procedural representation of some knowledge via mental simulation or projection of hypothetical events. Yet, at the very least, this thesis presents an existence proof that stands in direct opposition to the claim that declarative representation of second-order knowledge is too difficult.

cesses such as those to be presented in Part Three can better map symptoms of failure to their underlying faults (causes of failure) in support of blame assignment. This chapter presents the representational component of our content theory based, in part, on explanation-pattern theory. In particular, it will illustrate the concepts by assembling representations for the five failure symptom types described in the previous chapter.

## 4.1 Epistemology and Ontology

To support effective explanation of reasoning failure, and therefore to support learning, it is necessary to represent the thought processes and conclusions that constitute the reasoning being explained. A large number of terms exist in the English language that concern mental activity. Although surface features of a language utterance are not equivalent to the processes or states that may or may not lie behind a given utterance, a number of English expressions point to interesting problems for declarative representations. A few “cognitively homogeneous” terms will be examined that generally refer only to the internal world of the reasoner, rather than the external world of physical objects and other persons.<sup>35</sup> Thus, this chapter will ignore non-cognitive mental states such as emotions (affect, e.g., fear and love). Rather, it will focus on more simple concepts such as think, forget, and imagine, although humans are likely to think thoughts about the external world, forget to perform actions in the world, and imagine what the physical world may be like. With such constraints, the hope is to insulate the task by avoiding consideration of the more complex terms that intertwine the internal and external worlds, and instead, attempt to sketch an ontology of mental representations and a vocabulary of the content of such representations.

Many cognitive vocabularies make a prominent distinction between mental states (as knowledge or belief) and mental mechanisms (as the mental events that process knowledge or information). For example, conceptual dependency (CD) theory (Schank, 1972, 1975) distinguishes between two sets of representations: primitive mental ACTs and mental CONCEPTUALIZATIONs upon which the ACTs operate. In addition, the theory proposes a number of causal links that connect members of one set with members of the other. With such building blocks, a representational language such as CD must be able to represent many process terms: think (about), remember, infer, realize and calculate; and numerous state terms: fact, belief, guess, doubt, and disbelief. This document will refer to the execution of any mental process (or arbitrarily long string of processes) by the generic term *Cognize*,<sup>36</sup> and to a CONCEPTUALIZATION simply by the term *State* or *Mental-State*. See Figure 19, “Preliminary partial ontology of mental terms,”<sup>37</sup> for an initial

---

35. Certainly the boundary between the two worlds is not a very clean line. Terms such as “speak” concern the manipulation of mental terms (e.g., concepts), but complicate the representation with details of expression, translation, interpretation and the physical means of conveyance.

sketch of a target ontological vocabulary for mental representations. If a reasoner is to understand its own reasoning and mental conditions in any substantial level of detail, it will require a semantic hierarchy containing representations for most of these cognitive terms.

In addition to the state-process dichotomy, IML theory subdivides process terms by function into mental events that involve memory and transfer of information and those that involve computation or inference.<sup>38</sup> We associate inferential processes with logical or hypothetical reasoning. Example terms include *Hypothesize*, *Speculate*, *Deduce*, *Corroborate*, and *Postulate*. However, these inferential terms also include those that receive little attention in the AI community (e.g., *Intuit*). In Figure 19, inferential processes are subdivided into those driven by deliberate goals for processing (*Calculate*) and those in which belief is either more incidental or less exact (*Realize*).

Until recently, examples of memory processes such as *remember*, *remind*, *recall*, *recognize*, and *forget* (but here, the lack of a process occurring) have been largely unexamined and without explicit representation. Especially in the context of case-based reasoning or any problem solving that depends on indexed memory hierarchies to support a performance task, understanding the operation of memory can be of benefit when learning (see also Leake, 1995, Fox & Leake, 1995a, 1995b, and Kennedy, 1995, for additional arguments in favor of this position). A system that is to adjust the organization of memory will have a better chance of success if it has knowledge of the function and operation of the (cognitive) device it is changing. Thus, for a system to understand and change its own memory effectively, it is important that the system be able to represent the memory processes explicitly.

## 4.2 Granularity of Representation

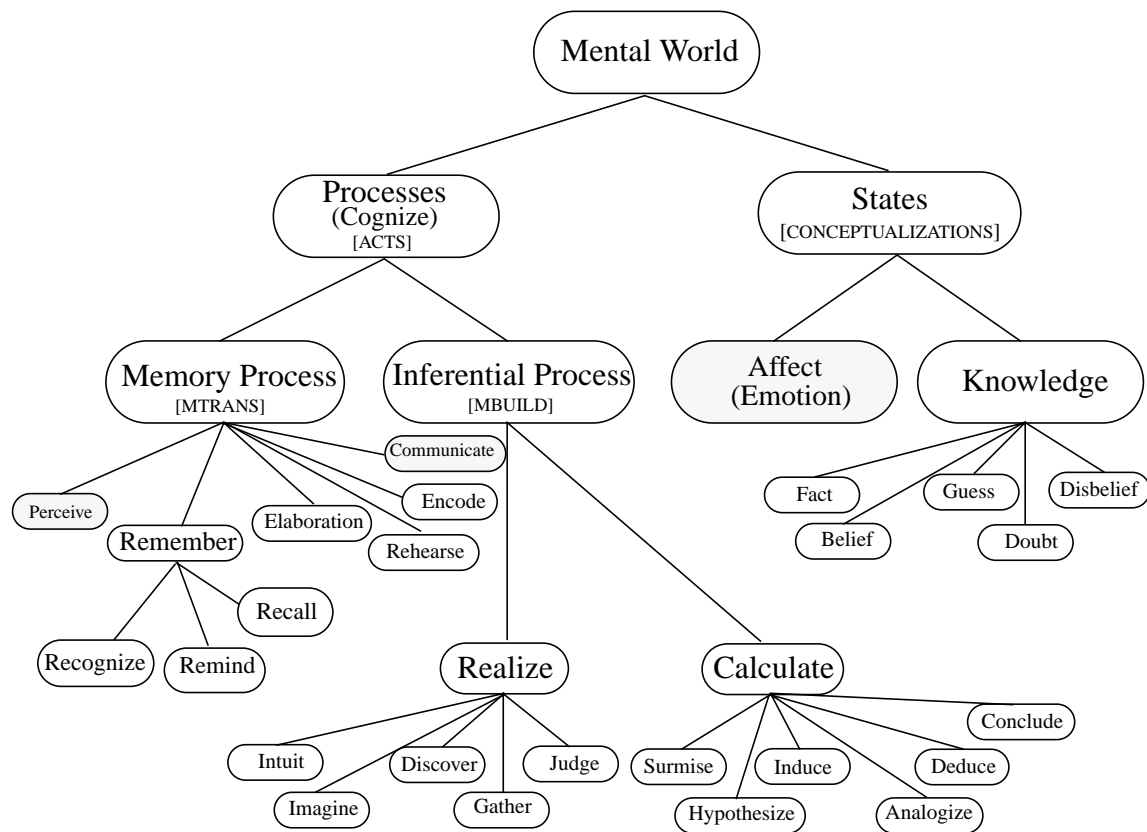
Schank (1975) developed CD theory to account for the kinds of inferences made by speakers of natural language. To represent language utterances, Schank composed a set of minimal conceptual primitives that would represent the interlingual basis that people used to reason when communicating, rather than simply some representation of the surface

---

36. The general term “to think” is an unfortunately overloaded term. It can be used in the sense of “to think about” (referring to a generic mental process) or “I think so” (referring to some qualitative level of confidence). Therefore, *cognize* is a less ambiguous representational label.

37. The shaded ovals in the figure represent unrepresented vocabulary terms. For example, this document ignores emotion (except for Descartes’ primitive passion taxonomy at the beginning of this chapter), either as a contributor to failure or as an ontological entity in need of representation.

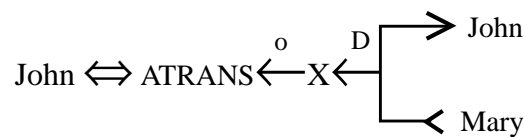
38. Schwanenflugel, Fabricius, Noyes, Bigler & Alexander (1994) analyzed folk theories of knowing. Subject responses during a similarity judgement task decomposed into memory, inference, and I/O clusters through factor analysis.




---

Figure 19. Preliminary partial ontology of mental terms

structure of language. As an example, ATRANS does not represent particular words, although it roughly maps to an abstract transfer of objects by an agent, such as the verb “to receive” at the surface level. The transfer of possession is not something that necessarily happens in the physical world (e.g., the transfer of ownership); rather it is an abstract social and psychological act that may or may not be accompanied by overt physical movements. Figure 20 shows an example of the ATRANS representation of “John received something from Mary.” The double arrow represents a two-way conceptual dependency relationship<sup>39</sup> between the agent on the left and the action on the right. Schank postulated a set of 11 primitive ACTs to which all language utterances by speakers on nontechnical topics could be mapped. The primitives have an important feature of canonical form; that is, all surface forms with the same meaning map to the same representation (e.g., “John received something from Mary.” is equivalent to “Mary gave something to John.”). They also provide a declarative structure and the semantic inferences (e.g., Mary’s object is no longer possessed by her) that provide meaning to the representation.




---

Figure 20. CD representation of abstract transfer (ATRANS)

o=physical object; D=direction

Yet, many have argued that such a small set of primitives are not sufficient to represent the meaning of many common natural language utterances. Wilensky (1986a) claims that many of the inferences made by the understanders of language are at an intermediate level of representation, rather than at a primitive level. For instance, the inference that “if a person buys a cake, the agent probably received it from a cashier” is found at the conceptual level of “buy,” rather than at the level of the ATRANS primitive in CD theory. Nothing in the meaning of abstract transfer is specific enough to include this condition.

Conversely though, to represent reflexive thoughts about reasoning, complete representations for all inferences and memory processes that generate such inferences, along with a complete enumeration of all knowledge dependencies, are not required. People cer-

---

39. A concept has a dependency relationship to a governing concept in a CD network when the dependent does not make sense without the governor and, moreover, when it further explains the governor (Schank & Tesler, 1969).

tainly cannot maintain a logically complete and consistent knowledge base, nor can they perform full dependency-directed backtracking (Stallman & Sussman, 1977) or reason maintenance for belief revision (Doyle, 1979); rather, they depend on failures of reasoning and memory of past errors to indicate where inconsistencies in their knowledge lie. That is, as knowledge is locally updated, a knowledge base will often become globally inconsistent and partially obsolete. It is at the point in which a system (either human or machine) attempts to reuse obsolete information that inconsistency becomes most apparent and further learning is enabled.<sup>40</sup> People often do know when they err if their conclusions contradict known facts, if plans go wrong, or if they forget (even if they cannot remember the forgotten item). Representations should support such types of self-knowledge, and it is at this level of granularity that an *epistemologically adequate* (McCarthy & Hayes, 1969) content theory of mental representations can be built.

For the above reasons, capturing the full level of details concerning mental activity is not necessary, and CD's two primitives mental ACTS are not sufficient to comprise an adequate representational system that can express states and mechanisms of the mental world. Rather, a vocabulary needs to be delivered that can minimally express the causal relationships involved in reasoning, but concurrently support the explanation of failure in sufficient detail that learning goals can be chosen. That is, granularity is determined functionally.

Q7: At what level of granularity should reasoning be represented?

Ans7: At enough detail to support learning from failure.

### 4.3 Representing Forgetting: An example

As an example of the kinds of representations that are required for effective introspective learning, this section will consider how to represent forgetting. The task is especially

---

40. See also McRoy (1993) for a related discussion of resource constraints on inference, the problems of logical inconsistency and logical omniscience, and the proposed relationship of these problems to the agent's own involvement in introspection. Related also, but from a psychological perspective, Glenberg, Wilkinson & Epstein (1982/1992) have shown that self-comprehension of text can be an illusion (i.e., people sometimes do not accurately monitor their own level of text comprehension), and they speculate that it is at the point where reading comprehension fails that humans are alerted to the need for learning.

interesting since the verb forget refers to a non-event, rather than a mental process. Issues will be addressed by considering three prevailing formalisms: logic, conceptual dependency, and explanation patterns. This section will show that, although all three representations have expressive ability, the explanation pattern knowledge representation possesses the most straight forward means for capturing the causal structure, inferences, and meaning of the mental term forget.

### 4.3.1 Logic

In order to use representations of mental terms effectively, a system should consider the structure of the representation, rather than simply how a system can syntactically manipulate representations or make sound inferences from them. In this regard, however, single logical predicates such as “forget” or “remember” are not entirely useful when trying to understand the memory failure of a person, P.

$$\begin{aligned} &\text{Forget (P, M)} \\ &\neg \text{Remember (P, M)} \end{aligned}$$

Because the predicates involve memory, it is helpful to posit the existence of two contrasting sets of axioms: the background knowledge (BK), or long-term memory of the agent, P, and the foreground knowledge (FK), representing the currently conscious or active axioms of the agent. Equation (8) shows the resulting interpretation of person P forgetting memory item M.

$$\text{Forget (P, M)} \rightarrow \exists M. (M \in \text{BK}_P) \wedge (M \notin \text{FK}_P) \quad (8)$$

With such a representation, one can also express the proposition that the person P knows that he has forgotten something; that is, the memory item, M, is on the tip of agent P’s tongue. P knows that M is in his background knowledge, but cannot retrieve it into his foreground knowledge:

$$\exists M. (M \in \text{BK}_P) \in \text{FK}_P \wedge (M \notin \text{FK}_P) \quad (9)$$

But to start to include these interpretations is to add content to the representation, rather than simply semantics. It is part of the *metaphysical interpretation* (McCarthy & Hayes, 1969) of the representation that determines an ontological category (i.e., what ought to be represented), and it begins to provide epistemological commitments (e.g., that the sets BK and FK are necessary representational distinctions). However, meaning is not only correspondences with the world to be represented, but meaning is also determined by the inferences a system can draw from a representation (Schank, 1975). The “forget” predicate

offers little in this regard. Moreover, this predicate will not assist a reasoning system to understand what happens when it forgets some memory item, *M*, nor will it help the system learn to avoid forgetting the item in the future. Finally, because the semantics of a mental event which did not actually occur is not represented well by a simple negation of a predicate representing an event which did occur (Cox & Ram, 1992a), the logical expression  $\neg \text{Remember}(\text{John}, M)$  is essentially a vacuous proposition. This is not to say that logic cannot represent such a mental “non-event,” rather, this simply illustrates that it is not an elementary task to construct an adequate representation of forgetting and that a single logical predicate will not suffice.

### 4.3.2 Conceptual Dependency

An alternative representational approach was undertaken by Schank, Goldman, Rieger & Riesbeck (1972) in order to specify the primitive representations for all verbs of thought in support of natural language understanding. They wished to represent what people say about the mental world, rather than represent all facets of a complex memory and reasoning model. They therefore used only two mental ACTS, MTRANS (mental transfer of information from one location to another) and MBUILD (mental building of conceptualizations), and a few support structures such as MLOC (mental locations, e.g., working memory, central processor and long-term memory).<sup>41</sup>

As a consequence, the CD representation of forgetting by Schank and his colleagues is as depicted in Figure 21 (cf. Figure 20). John does not mentally transfer a copy of the mental object, *M*, from the recipient case, that of John’s long-term memory, to his central processor. Such a representation does provide more structure than the predicate forms above, and it supports inference (e.g., if *M* was an intention to do some action, as opposed to a proposition, then the result of such an act was not obtained; Schank, 1975, p. 60). However, the CD formalism cannot distinguish between the case during which John forgot due to *M* not being in his long-term memory and a case of forgetting due to missing associations between cues in the environment and the indexes with which *M* was encoded in memory. Thus, it does not provide enough information to learn from the experience.

---

41. Schank, Goldman, Rieger, & Riesbeck (1972) actually referred to working memory as immediate memory and the central processor as a conceptual processor. I have used some license to keep terms in a contemporary language. Moreover, Schank and his colleagues used a third primitive ACT, CONC, which was to conceptualize or think about without building a new conceptualization, but Schank (1975) eliminated it from the theory. For the purposes of this research, however, the differences do not matter.



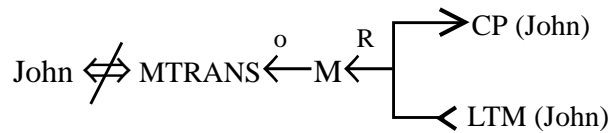


Figure 21. CD representation of forgetting

o=mental object or conceptualization; R=Recipient;  
CP=Central Processor; LTM=Long Term Memory

### 4.3.3 Explanation Pattern Theory

To represents some of the nuances implied by the term forget and not easily captured by either logic or CDs, IML theory uses an extension of Explanation Pattern (XP) theory (Leake, 1992; Owens, 1990a; Ram, 1989, 1991, 1994; Schank, 1986; Schank & Kass, 1990).<sup>42</sup> Essentially, XPs are directed graphs with nodes that are either states or processes and links that are either ENABLES links (connecting states with the processes for which they are preconditions), RESULTS links (connecting a process with a result), or INITIATE links (connecting two states). The links of an XP (as we use them here) include numbering to indicate relative temporal sequence between links. Figure 22<sup>43</sup> illustrates an explanation for why a volitional agent, A, performs a given action, M (i.e., it explains the actor relation of M). The causal reason is that the agent has the goal of achieving the desired state, GS, and the agent knows that GS will results from M if he performs it (Ram, 1989).

A Meta-XP is similar to a standard XP in that it is an explanatory causal structure. The major difference between the two is that instead of presenting a causal justification for a physical relation (such as why people look like their ancestors) or a volitional role relation (such as why a person performs a given action), a Meta-XP explains how and why an agent reasons in a particular manner.

The Meta-XP structure of Figure 23 represents a memory retrieval attempt enabled by goal, G, and context cues, C, that tried to retrieve some memory object, M, given an index, I, that did not result in an expectation (or interpretation), E, that should have been equal to

42. See also Almonayyes (1994), Kerner (1995), and Schank, Kass & Riesbeck (1994) for additional applications of XP theory.

43. Attributes and relations are represented explicitly in these graphs. Thus, the ACTOR attribute of an event X with some value Y is equivalent to the relation ACTOR having domain X and co-domain Y. Section 4.4 provides further details concerning this notation.

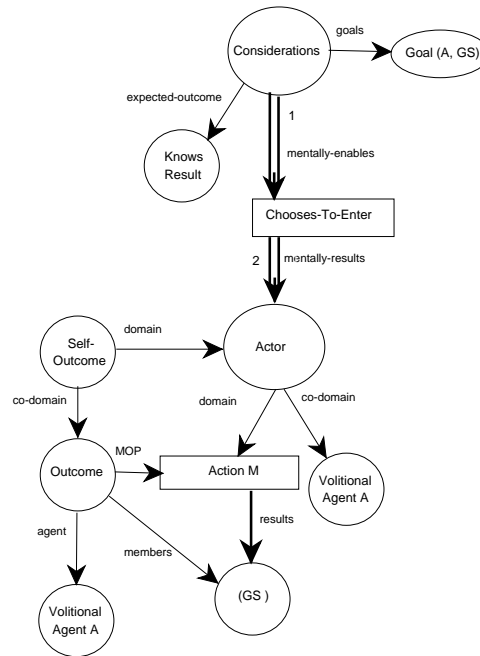


Figure 22. XP representation of XP-GOAL-OF-OUTCOME->ACTOR  
 GS=good state; MOP=memory organization package

some actual item, A. The fact that E is out of the set of beliefs with respect to the reasoner's foreground knowledge (i.e., is not present in working memory) initiates the knowledge that a retrieval failure had occurred.

Because forgetting is not a mental event, but rather the lack of successful memory processing, challenges exist when representing it. Forgetting can be expressed properly only if a system can represent that it does not believe a successful memory retrieval has occurred. The belief logic of Doyle (1979) has four truth values for a given proposition "p." If p is believed then it is in the set of beliefs, whereas if p is not believed then it is out. Conversely, the negation of the assertion of p may be either in or out of the agent's set of beliefs. Therefore, the four truth values are  $p(in)$ ,  $p(out)$ ,  $\neg p(in)$ , and  $\neg p(out)$ . Using these values, a system needs to be able to declare that there is a memory item that was not retrieved.

The system could create a dummy concept representing the forgotten item that it believes did not result from some retrieval process. This concept could be marked as disbelieved with the above notation, since it was not retrieved and cannot be specified by the system. But technically, it is incorrect to assert that the concept is not believed, if it is in the system's background knowledge. In other words, it is believed but not recalled. Cox & Ram's (1992a) response to this dilemma was first, to assume a special set of beliefs rep-

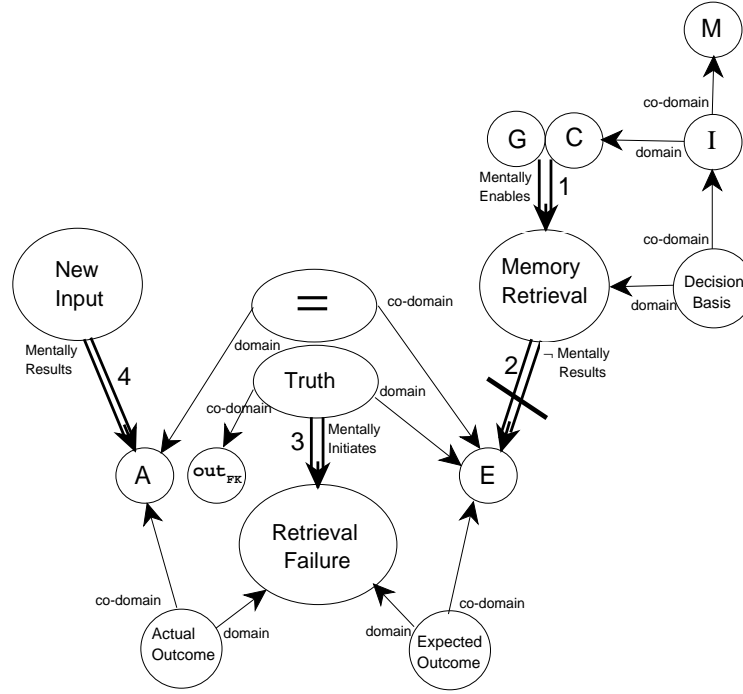


Figure 23. Meta-XP representation of forgetting  
A=actual; E=expected; G=goal; C=cues; M=memory item; I=memory index.

representing the working memory of the system (the FK), and then secondly, to modify Doyle's belief logic to claim belief membership with respect to a particular set of beliefs. Thus, P, a given memory item that was not retrieved, may be in the set of beliefs with respect to the BK, written  $P(in_{BK})$ , but out of the set of beliefs with respect to the FK, written  $P(out_{FK})$ .<sup>44</sup>

As specified in Table 6, the number of ways that the memory retrieval process may fail depends on the conditions of the nodes A, E, G, I, and M. If the memory item, M, is not in the BK (i.e.,  $M(out_{BK})$ ), then there is nothing to be retrieved. This can occur either because there never was a concept in memory to be retrieved, or because the item was previously deleted from memory. Ostensibly, no difference exists between the two in this representation. For example, in the Meta-AQUA story understanding system, a novel situation exists when trying to explain a police dog barking at a passenger's luggage in the airport (Section 2.1.1). The system had previously encountered dogs barking only at animate

44. Compare this with the assumption maintenance system discussed by McDermott (1989). In general, propositions may be *in* or *out* with respect to arbitrary sets of beliefs, which in the Meta-AQUA system are used to represent what is in the FK during different reasoning experiences.

Table 6: Truth values of graph nodes for forgetting

Description	A	E	G	I	M
Absent Memory (Novel Situation)	$in_{FK}$	$out_{FK}$	$in_{FK}$	$out_{BK}$	$out_{BK}$
Absent Index (Missing Association)	$in_{FK}$	$out_{FK}$	$in_{FK}$	$out_{BK}$	$in_{BK}$
Absent Retrieval Goal	$in_{FK}$	$out_{FK}$	$out_{FK}$	$\emptyset$	$\emptyset$
Absent Feedback	$out_{FK}$	$out_{FK}$	$\emptyset$	$\emptyset$	$\emptyset$

$\emptyset$  = don't care

objects, so it had no structure in memory to understand this novel event. Although this example does not represent forgetting *per se*, in systems that delete memory items in order to facilitate learning (e.g., Markovitch and Scott, 1988; Smyth & Keane, 1995), trying to retrieve a deleted item is equivalent to a novel situation from a computational perspective.

A more prototypical instance of forgetting is illustrated in the second Meta-AQUA story of Chapter II (Section 2.1.2). The system acquires a new explanation for why dogs bark, but because it indexes it by containers, the subsequent story is not able to retrieve it given the context of a dog barking at a pile of dirty clothes (the location of the suspect's stash of contraband). The correct index is missing and so no explanation is retrieved, although it certainly knows the correct explanation. The node truth values on the Meta-XP representation of this event are arrayed according to the second row from the top of Table 6 (refer to Figure 23 for the corresponding nodes or peek ahead to Figure 29 on page 88).

These tabularized values on the representation of Figure 23 capture an entire class of memory failures: failure due to a missing index, I; failure due to a missing object, M; failure because of a missing retrieval goal, G;<sup>45</sup> or failure due to not attending to the proper cues, C, in the environment. Having such a declarative representation allows the system to reason

---

45. A missing memory-retrieval goal is equivalent to an agent never trying to remember. For instance, the reasoner may have wanted to ask a question after a lecture was complete, but failed to do so because he never generated a goal to remember once the lecture was complete. Alternatively the agent may know at the end of the lecture that he needs to ask something, but cannot remember what it was. This second example is the case of a missing index. Note that both a missing index and an incorrect index may be present at the same time. In such cases, a target item is not retrieved, whereas an interfering item is retrieved instead.

about the various causes of forgetting; it can inspect the structural representation for a memory failure and therefore, analyze and consider the reasons for the memory failure. Such an ability facilitates learning because it allows a learner to explain the reasoning failure and use the result in determining what needs to be learned.<sup>46</sup>

## 4.4 Meta-Explanation Patterns

In most interpretations (e.g., Kuokka, 1990, p. 5; Hayes-Roth, Waterman, & Lenat, 1983, p. 402), meta-X can be translated to “X about X. Therefore, a meta-explanation pattern (Meta-XP) is an explanation pattern about another explanation pattern. Whereas, an Explanation Pattern (XP) is a causal structure that explains a physical state by presenting the chain of physical events causing such states, a Meta-XP is an explanation of how or why an XP is generated incorrectly or otherwise fails.<sup>47</sup> Two classes of Meta-XPs exist to facilitate a system’s ability to reason about itself and to assist in selecting a learning algorithm or strategy. A *Trace Meta-XP* (TMXP) explains *how* a system generates an explanation about the world (or itself), and an *Introspective Meta-XP* (IMXP) explains *why* the reasoning captured in a TMXP goes awry. Thus, a TMXP records the extent of reasoning tasks and the reasons for decisions taken during processing. An IMXP is a general causal structure that represent explanations of various the failure types from the taxonomy of Chapter III. The IMXP structures represent past experience of reasoning about the self (i.e., cases of meta-reasoning) and assist in forming the learning goals of the system after failure occurs. Whereas a TMXP records the immediate mental events of the reasoner and they exist in the FK, IMXPs are retrieved from the BK and applied to TMXPs. This case-based approach to self understanding is similar to the operations by which standard XPs are retrieved and applied to input representations for story understanding. The same basic algorithm is used in each.

Q5: How to represent mental states and reasoning mechanisms?

Ans5: Use meta-explanation patterns.

46. See Cox (1994b) for a discussion of related computational models of forgetting.

47. Here the definition of a Meta-Explanation is interpreted in a narrow sense as applied to understanding tasks involving the explanation of anomalies. In general, however, a Meta-XP may be any explanation of how and why an agent reasons in any particular way, including processes other than explanation.

Explanation patterns (XPs) are similar to justification trees, linking antecedent conditions to their consequences. The XP is essentially a directed graph of concepts, connected with RESULTS, ENABLES and INITIATES links. A RESULTS link connects a process with a state, while an ENABLES link connects a precondition state to a process. An INITIATES link connects two states. The set of sink nodes in the graph is called the PRE-XP-NODES (see Figure 24). These nodes represent what must be present in the current situation for the XP to apply. One distinguished node in this set is called the EXPLAINS node. It is bound to the concept which is being explained. Source nodes are termed XP-ASSERTED-NODES. All other nodes are INTERNAL-XP-NODES.

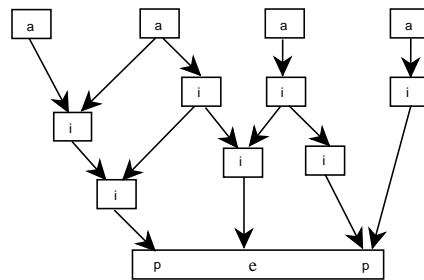


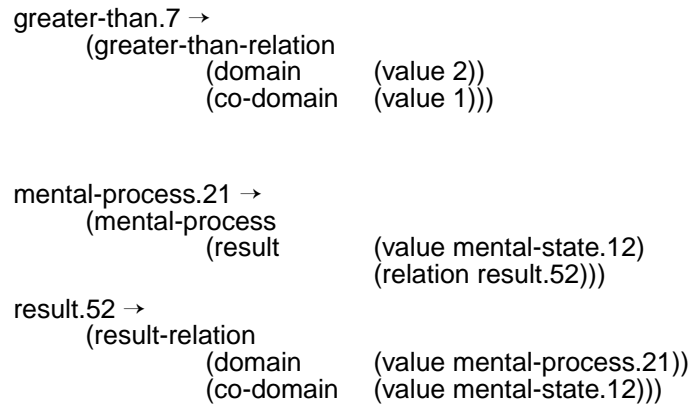
Figure 24. XP as a directed graph  
a=asserted; i=internal; p=pre-XP; e=explains

For an XP to apply to a given situation, all PRE-XP-NODES must be in the current set of beliefs. If they are not, then the explanation is not appropriate to the situation. If the structure is not rejected, then all XP-ASSERTED-NODES are checked. For each XP-ASSERTED node verified, all INTERNAL-XP-NODES connected to it are verified. If all XP-ASSERTED-NODES can be verified, then the entire explanation is verified.<sup>48</sup>

In the representation presented here, attributes are treated as first-class objects; that is, attribute relations have an explicit frame representation. For instance, the greater-than relation of Figure 25 has both domain and co-domain slots. Therefore, the token greater-than.7 expresses the proposition that the integer two is greater than one. Moreover, this same notation can represent the slot (attribute) of a frame. The result attribute of mental-process.21 is a relation (result.52) from its domain (mental-process.21) to its co-domain (mental-state.12). As indicated by the lowest level of parentheses, frame slots have both value and relation facets. This explicit representation allows a system to assert specific propositions about slots, instead of only values. A system can therefore ask a question about the result slot itself. Questions such as “What was the result of mental-pro-

48. See Ram (1994) for additional details concerning the structure and use of explanation patterns.

cess.21?” need only refer to the value facet of the attribute; but, questions such as “Why did the process result in state.12?” can only be formed properly if the `result` relation has an explicit representation (Ram, 1989; Wilensky, 1986b).




---

Figure 25. Relations as first-class objects  
arrows = token assignments

#### 4.4.1 Trace Meta-XPs

Ram (1990, 1994) has developed a theory of motivational explanation based on decision models which characterize the decision process that an agent goes through in deciding to perform an action. For example, the religious-fanatic explanation for suicide bombing is a decision model describing why a bomber would choose to perform a terrorist strike in which the bomber dies (see Figure 26 and compare to Figure 22 on page 76).<sup>49</sup> Ram's model claims that an agent first considers any relevant goals, goal priorities, and the expected outcome of performing the action. The actor then makes a decision whether or not to enter into such a role, and if so, performs the action. IML theory extends the model to account for introspective reasoning.

Meta-reasoning can be conceptualized in a similar manner. A set of states, priorities, and the expected strategy outcome determine a reasoner's decision of a processing strategy, like the above factors determine the actor's decision to act. Based on general knowledge, current representation of the story, and any inferences that can be drawn from this knowledge, the reasoner chooses a particular reasoning strategy. Once executed, a strategy may

---

49. The Considerations (two goals and one belief) comprise the XP-ASSERTED-NODES, the Chooses-To-Enter node is the lone INTERNAL-XP-NODE, and the Actor relation is the only PRE-XP-NODE (and represents the EXPLAINS node of the XP).

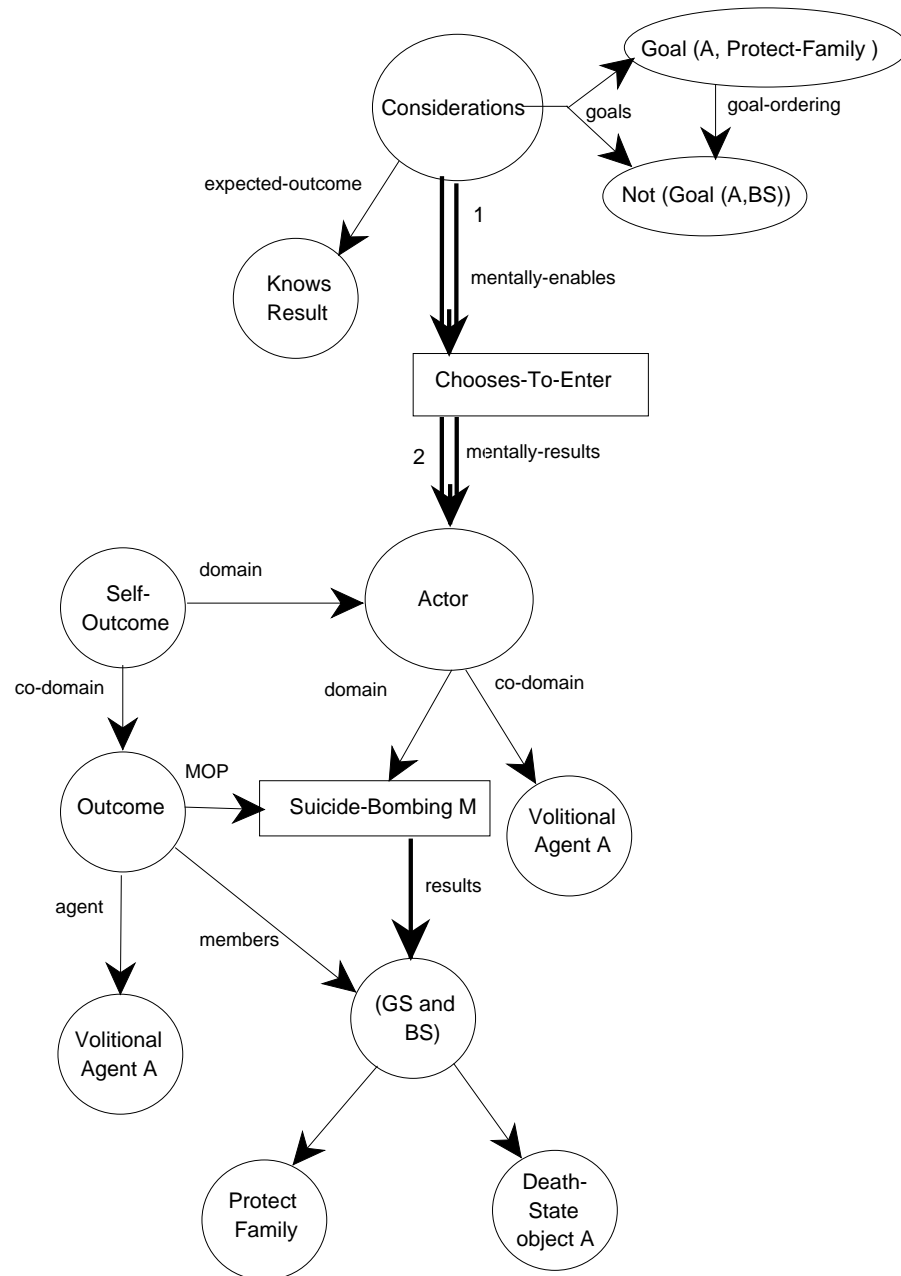


Figure 26. Volitional XP for why agent performs suicide bombing (adapted from Ram, 1993)

BS=bad state; GS=good state; MOP=memory organization package



produce further reasoning, requiring additional strategy decisions.

These decisions are chained into threads of reasoning such that each one initiates the goal that drives the next. Though the chains can vary widely, in the task of question-driven story understanding, the chains take the following form: Identify Anomaly → Generate Explanation → Verify Hypothesis (see Figure 14 on page 36 in Chapter II). Note that since the explanation generation phase produces a hypothesis and the verification phase produces a measure of goodness, if the hypothesis has been confirmed with a sufficiently high confidence, then the overall product of the understanding process has been a sound explanation. Alternatively, if the explanation has been disconfirmed, then a later failure identification phase should generate the question “Why did the explanation fail?” This knowledge goal triggers the learning process.

The understanding process is recursive in nature. For example, if a hypothesis generates a new question, then the reasoner will spawn a recursive regeneration of the sequence because an unanswered question is anomalous. Like physical explanations that explain how objects work in the physical world, and volitional explanations that explain why agents perform various acts in the world, introspective explanations explain how and why conclusions are drawn by the reasoner; they explain events in the mental world.

When insufficient knowledge exists on which to base a decision, a useful strategy is to simply defer making the decision. The reasoning task is suspended and later continued if and when the requisite knowledge appears. This is a form of opportunistic reasoning. Meta-XPs are able represent chains of reasoning that follow from opportunistic reasoning as well as uninterrupted decisions.

A Trace Meta-XP, representing the trace of the reasoning process, is a chain of *Decide-Compute-Nodes* (D-C-Nodes). Figure 28 shows the outermost frame definition<sup>50</sup> of the decide-compute-node type whose graph structure is shown in Figure 27. A non-recursive single instance of explanation would be a chain of three D-C-Nodes, one for each phase in the anomaly-identification/generate-explanation/verify-hypothesis sequence.<sup>51</sup> These nodes record the processes that formulate the knowledge goals of a system, together

---

50. Frame figures use the following notation:

X.0 - - - - - an attribute value;  
 (X) - - - - - a frame of type X;  
 =X - - - - - variable binding to the outermost slot named X;  
 (=X =Y) - - a list of variable bindings;  
 (X =Y) - - - an frame of type X with referent alias named Y.

This last feature is included so that variable can be bound to slots other than the outermost slots.

with the reasons for and the results and side-effects of performing such mental actions. The trace of reasoning is similar to a derivational analogy trace as described by Carbonell (1986) and Veloso and Carbonell (1994). A Trace Meta-XP is a specific explanation of why a reasoner chooses a particular reasoning method and what results from the strategy. Like an XP, the Meta-XP can be a general structure applied to a wide range of contexts, or a specific instantiation that records a particular thought process. One distinguishing property of Trace Meta-XPs is that a decision at one stage is often based on features in previous stages. For example, the decision of how to verify a hypothesis may be based on knowledge used to construct the hypothesis initially. This property, deciding based on previous knowledge, is particularly true of learning, which, by definition, is based on prior processing.

#### 4.4.2 Introspective Meta-XPs

Whereas a Trace Meta-XP explains how a failure occurred, by providing the sequence of mental events and states along with the causal linkage between them, an Introspective Meta-XP explains why the results of a chain of reasoning are wrong. The IMXP posits a causal reckoning between the events and states of the TMXP. In addition, an IMXP provides a learning goal specifying what needs to be learned. Then, given such an explanation bound to a reasoning chain, the task of the system is to select a learning strategy to reduce the likelihood of repeating the failure.

An IMXP consists of six distinctive parts:

- The IMXP type class.
- The failure type accounted for by the IMXP.
- A graph representation of the failure.
- Temporal ordering on the links of the graph.
- An ordered list of likely locations in the graph where processing errors may have occurred.
- A corresponding list of learning goals to be spawned for failure repair.

There are three classes of IMXPs: base, core, and composite. *Base* types constitute the basic vocabulary (labels) with which *core* IMXPs are built. We have identified seven primitive types in the base class: successful prediction, inferential expectation failure,

---

51. Note that in most of this work the initial phase of anomaly identification is simplified. Rather than considering all four steps represented in a D-C-Node, the algorithm skips the input analysis step, posts a goal to interpret the input, and then uses only a single strategy as outlined in the text. The result is a signal whether or not an anomaly exists together with the anomaly's cause. See, for example, Figure 14, "Question-driven understanding," on page 36.

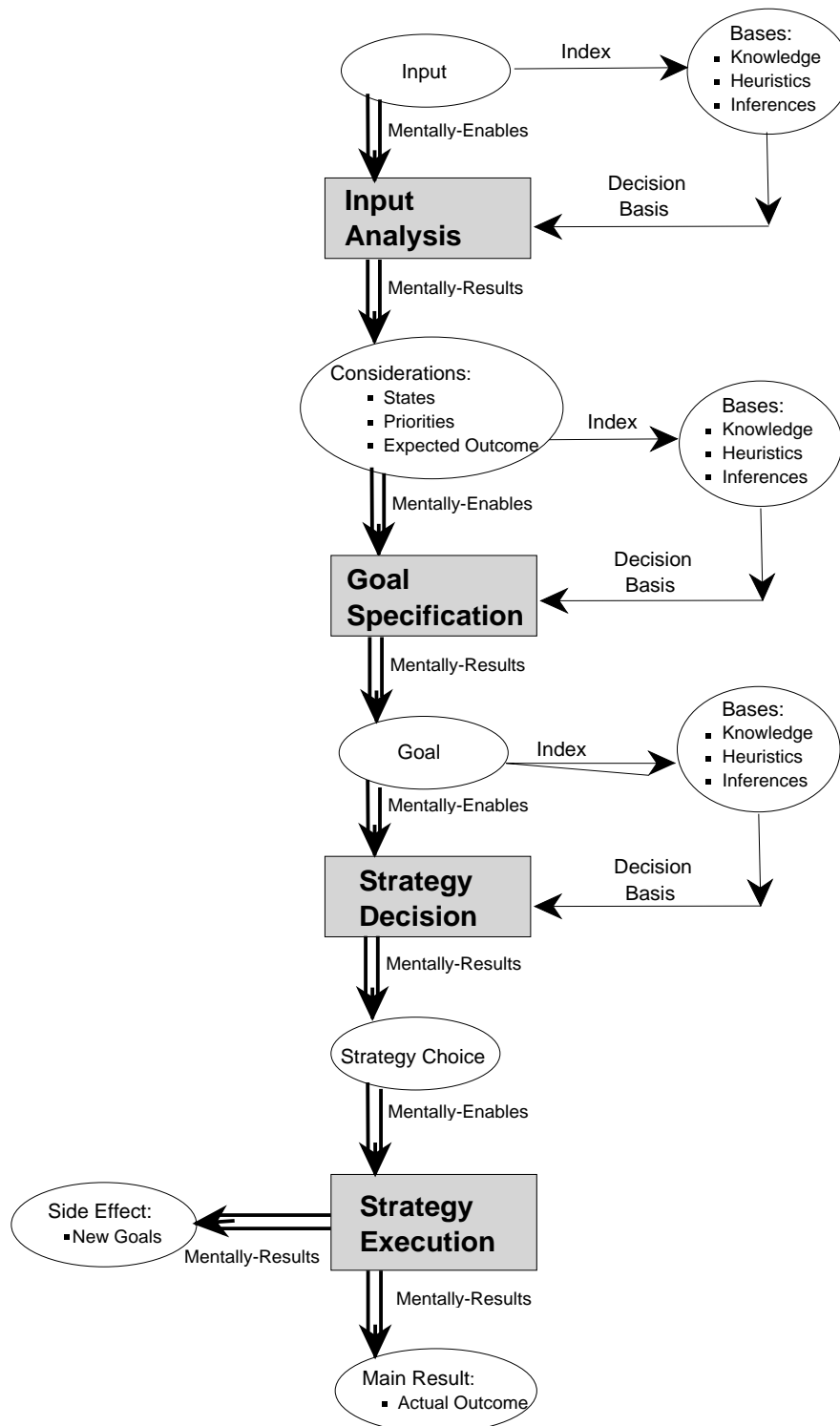


Figure 27. Graph structure for Decide-Compute-Node

```

(define-frame D-C-NODE
  (isa      (value (meta-xp)))
  (actor    (value (volitional-agent)))           ;; The agent making the decision.
  (enables- (value (d-c-node)))                   ;; Points to previous d-c-node in TMXP chain.
  (initial-state (value (considerations)))         ;; Preconditions
  (strategy-choice
    (value (strategy-choice-value)))              ;; What strategy chosen
  (strategy-decision
    (value (decision-process
      (actor (value =actor)
        (relation =role))                         ;; Same as role slot below.
      (basis-of-decision (value (basis)))
      (main-result (value =strategy-choice))))))
  (role      (value (actor
    (domain (value =strategy-decision))
    (co-domain (value =actor))))))               ;; The actor slot of the decision-process.
  (strategy-execution
    (value (mental-process)))
  (side-effect (value (considerations)))
  (main-result (value (outcome
    (results-
      (value =strategy-execution))))))           ;; Return values from the strategy-execution.
                                                    ;; Backpointer from where this outcome resulted
  (explains    (value =role))                     ;; Explains why agent picks strategy-execution
  (pre-xp-nodes (value (=explains =main-result =side-effect)))
  (xp-asserted-nodes
    (value (=initial-state)))
  (internal-nodes
    (value (=strategy-decision =strategy-choice
      =strategy-execution)))
  (link1      (value (mentally-enables
    (domain (value =initial-state))
    (co-domain (value =strategy-decision))))))
  (link2      (value (mentally-results
    (domain (value =strategy-decision))
    (co-domain (value =strategy-choice))))))
  (link3      (value (mentally-enables
    (domain (value =strategy-choice))
    (co-domain (value =strategy-execution))))))
  (link4      (value (mentally-results
    (domain (value =strategy-execution))
    (co-domain (value =main-result))))))
  (link5      (value (mentally-results
    (domain (value =strategy-execution))
    (co-domain (value =side-effect))))))
  (links      (value (=link1 =link2 =link3 =link4 =link5))) ;; List of all causal links.
)

```

---

Figure 28. Frame definition for Decide-Compute-Node

incorporation failure, belated prediction, retrieval failure, construction failure, and input failure. The core types are representations of the failure types enumerated in Table 4, “Final table for reasoning model,” on page 50. They include representations for failures such as contradiction and impasse, and the IMXP representation for each will be shown in Section 4.7. Core types are combined to form *composite* IMXPs that describe situations encountered by reasoning agents, such as the “Drug Bust” examples in Section 2.1.

The internal graph structure of an IMXP consists of nodes, representing both mental states and mental events (processes), and the causal links between them. The nodes and links have the same semantics as those described for TMXPs in section 4.4.1. The graph gives both a structural and a causal accounting of what happened and what should have happened when information was processed.

Consider the graph diagram in Figure 29 (cf. Figure 23). It represents the introspective reasoning of the second drug-bust story in Chapter II (Section 2.1.2). In this story, the Meta-AQUA system forgets the explanation learned in the previous story, that dogs will bark at inanimate objects when they detect contraband. Because this explanation was indexed by containers, the system retrieves no explanation to explain why the dog is barking at a pile of dirty clothes; that is, it experiences a memory impasse. Later in the story, when the officer praises the dog for barking at the clothes, the system infers that the explanation should have been a detection explanation. This graph structure represents the composite IMXP ANOMALY-AND-BAFFLED. It contains but one core case, a missing association, and has at its heart the base case of retrieval failure. In Figure 30, a frame definition is provided for the IMXP composite type from which the instance portrayed in Figure 29 was formed.<sup>52</sup>

Base class IMXPs represent a primitive type or component in the content theory of mental events from which traces of reasoning failures may be constructed. The goal is to enumerate a sufficient number of these basic types to cover the major kinds of reasoning failures that arise in story understanding and other tasks. The types of failures (discussed Section 3.2 on page 45) fall into two complementary classes: commission error and omission error. Commission errors stem from reasoning which should not have been performed or knowledge which should not have been used. Omission errors originate from the lack of some reasoning or knowledge. The content theory herein contains Base IMXPs to describe both classes of failure.

---

52. The frame definition is simplified in order to show it here. All facet notation is removed because only value facets of slots are shown in the figure. In addition, the figure shows only the important slots to illustrate the definition. Some slots are missing.

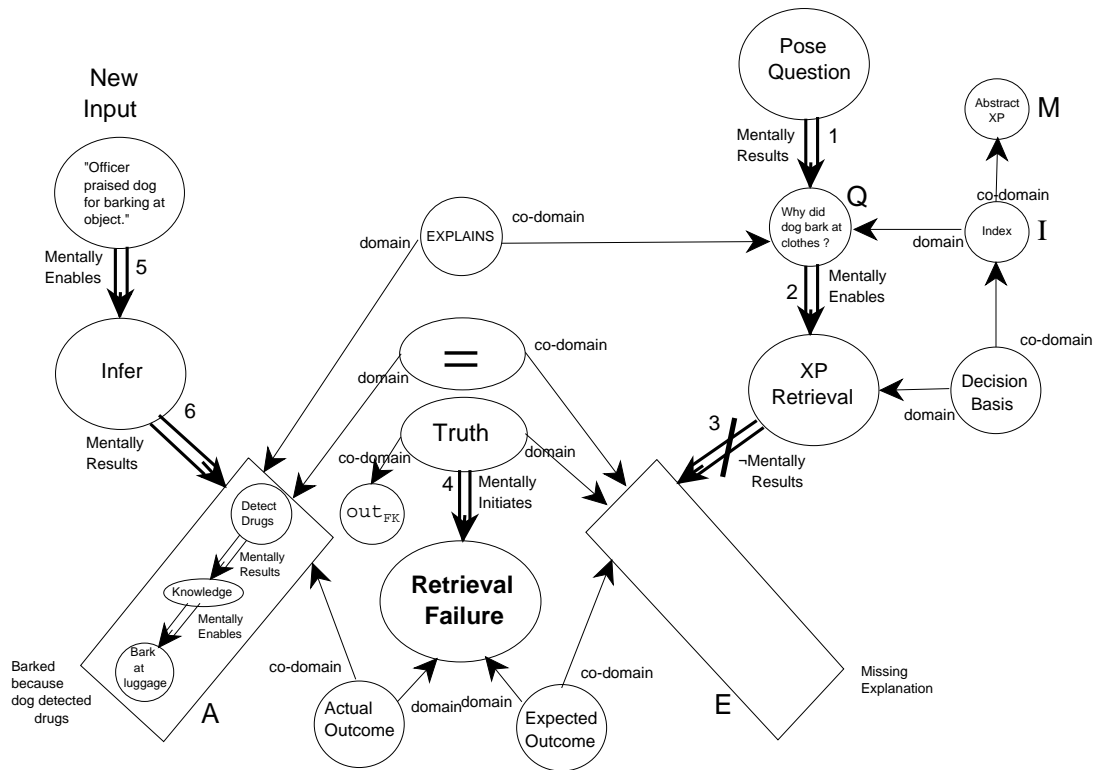


Figure 29. Representation for forgotten detection explanation  
 A=actual; E=expected; Q=question; I=index; M=memory item

```

(define-frame IMXP-BAFFLED-AND-RESOLVED
  (isa      (composite-introspective-meta-xp))           ;; IMXP Class
  (failure-cause (novel-situation.0 missing-assoc.0))    ;; Which one we do not know
  (q        (relation(explanations (=a))))              ;; Baffling question
  (a        (xp      (explains =q)))                     ;; Actual explanation
  (e        (xp      (explains =q)))                     ;; Missing expectation
  (i        (index (domain =q) (co-domain =m)))          ;; Index used to retrieve E
  (m        (xp))                                         ;; Forgotten xp.
  (truth-value (truth (domain =e)                        ;; E not in set of beliefs wrt FK
                     (co-domain out-fk.0)))
  (equals     (equal-relation(domain =a)                 ;; Actual should have been
               (co-domain =e)))                          ;; equal to what was expected
  (rf         (retrieval-failure(initiates- =truth-value) ;; The memory failure
               (expected-outcome =e)                      ;; explained by the IMXP
               (actual-outcome =a)))
  (new-input  (entity))                                   ;; Story input
  (later-process (cognize))                               ;; Inference in this case
  (rc         (trace-meta-xp                             ;; Reasoning chain
               (identification =q-id)
               (generation =hypo-gen)
               (link3 =link2)
               (link4 (mentally-results (truth out-fk.0)))))
  (q-id       (d-c-node                                   ;; Question identification
               (strategy-choice questioning.0)
               (strategy-execution pose-question.0)
               (side-effect (considerations =con
                             (prime-state =k-goal)))
               (link4 =link1)))
  (k-goal     (knowledge-acquisition-goal                 ;; Knowledge goal to answer
               (goal-object                               ;; the question
                (generate (co-domain =q))))))
  (hypo-gen   (d-c-node                                   ;; Hypothesis generation
               (strategy-decision =h-decision)
               (main-result (outcome =o (members (=link4))))
               (link4 (mentally-results (co-domain =o)
                                         (truth out-fk.0)))))
  (h-decision (decision-process                           ;; XP retrieval in this case
               (basis-of-decision =h-decision-basis)))
  (h-decision-basis
    (basis (knowledge                                     ;; Existence of I is the basis
            (collection                                  ;; to use case-based explanation
             (members ((knowledge-state
                        (co-domain =i)
                        (believed-item =i)))))))
  (links      (=link1 =link2 =link3 =link4 =link5 =link6)) ;; Links are in temporal order
  (link1      (mentally-results (domain pose-question.0)
               (co-domain (outcome (members (=q))))))
  (link2      (mentally-enables (domain =con)
               (co-domain =hypo-gen)))
  (link3      (mentally-results (domain =rc)              ;; and all correspond to the
               (co-domain =e)))                          ;; numbered links in Figure 29.
  (link4      (mentally-initiates (domain =truth-value)
               (co-domain =rf)))
  (link5      (mentally-enables (domain =new-input)
               (co-domain =later-process)))
  (link6      (mentally-results (domain =later-process)
               (co-domain =a)))
  (explains    =rf)                                       ;; What the IMXP explains.
  (pre-xp-nodes (=a =e =rf))                             ;; XP consequences.
  (internal-nodes (=q =hypo-gen =later-process =i))       ;; Neither sink nor source nodes
  (xp-asserted-nodes (=q-id =m =new-input))               ;; XP antecedents.
  (potential-faults (=a =i))                             ;; Nodes for blame-assignment
  (potential-learning-goals                               ;; Corresponding learning goals
    (knowledge-expansion-goal
     (goal-object =a)                                     ;; Expand the new explanation
     (subgoals =krg)
     (priority (integer-value =pr))
     (backptr (plan))
     (conditions ((inferred.0 acquired.0)))
     (knowledge-reorganization-goal =krg
      (goal-object =i)                                   ;; Reorganize memory to hold it
      (priority (integer-value (less-than =pr))))))

```

Figure 30. IMXP frame definition for forgetting

## 4.5 Vocabulary

The partial ontological hierarchy of mental terms in Figure 19 on page 70, pictures some basic type identifiers of the mental domain. They represent the most fundamental labels used to identify particular classes of mental actions and states and provide the primitive building blocks with which declarative structures are assembled to describe the processing that occurs within intelligent systems. A major goal of an understanding system operating in a mental world is to refine the labels of structures as additional knowledge is gained about particular actions in the domain. For instance, a system may only know that a particular node is some kind of cognitive process, thus it labels it with the vocabulary term *Cognize*. If the system subsequently discovers that it is a memory process, the label can be refined to *Memory Process*.<sup>53</sup> As more information is ascertained, the system may determine that the structure actually represents *Recall* or *Recognize*. As each label is refined, additional inferences are warranted.

Not shown in Figure 19 are the terms used to represent failure. These vocabulary labels are the base IMXP types mentioned in the previous subsection. This research has identified two types of commission error labels: *Inferential expectation failures* typify errors of projection. They occur when the reasoner expects an event to happen in a certain way, but the actual event is different or missing. *Incorporation failures* result from an object or event having some attribute that contradicts some restriction on its values. Four omission error labels have also been identified: *Belated prediction* occurs after the fact. Some prediction that should have occurred did not, but only in hindsight is this observation made. *Retrieval failures* occur when a reasoner cannot remember an appropriate piece of knowledge; in essence, it represents forgetting or memory failure. *Construction failure* is similar, but occurs when a reasoner cannot infer or construct a solution to a problem. *Input failure* is error due to lack of some input information. To construct the five core types of failure (outlined in Section 3.2), these labels are used. The basic organization for all of these representations is at the level of a comparison between an expectation and some feedback (either from the environment or additional inference or memory).<sup>54</sup> Oehlmann, Edwards, and Sleeman (1995) stress the importance of metacognitive processing to provide expectations and to monitor comprehension, both in human and machine systems. The representations used by any system should support these processes. The following sections provide representations for both successful and for failed mental processing.

---

53. Although the vocabulary item is listed as *Memory Process* in Figure 19, the IMXP figures have been using *Memory Retrieval*. The terms are used interchangeably.

54. See Krulwich (1995) for another view on basic level categories for mental representations. Instead of granularity, however, his discussion centers on the proper level of abstraction.





```

(define-frame IMXP-SUCCESSFUL-PREDICTION
  (isa      (core-introspective-meta-xp))
  (failure-cause nil.0)
  (a      (entity))
  (e      (entity)
    (truth hypothesized-in.0)))
  (g      (goal))
  (rc      (trace-meta-xp
    (identification
      (d-c-node =d-c-n
        (initial-state =preconditions) ))
      (main-result =e))))
  (preconditions (considerations
    (mentally-enables =d=c=n)
    (prime-state =g)))
  (comparison (inferential-process
    (arg1 =a)
    (arg2 =e)
    (main-result =equals)))
  (sp      (successful-prediction
    (initiates- =equals)
    (expected-outcome =e)
    (actual-outcome =a)))
  (equals   (equal-relation
    (domain =a)
    (co-domain =e)))
  (later-process (cognize))
  (nodes      (=a =e =sp =equals =rc =later-process
    =comparison =preconditions))
  (pre-xp-nodes (=a =e =sp))
  (explains    =sp)
  (internal-nodes (=equals =rc))
  (xp-asserted-nodes
    (=later-process =comparison =preconditions))
  (link1      (mentally-enables
    (domain =preconditions)
    (co-domain =rc)))
  (link2      (mentally-results
    (domain =rc)
    (co-domain =e)))
  (link3      (mentally-results
    (domain =later-process)
    (co-domain =a)))
  (link4      (mentally-results
    (domain =comparison)
    (co-domain =equals)))
  (link5      (mentally-initiates
    (domain =equals)
    (co-domain =sp)))
  (links      (=link1 =link2 =link3 =link4 =link5))
)

```

;; No failure.  
 ;; Actual outcome.  
 ;; Expected outcome.  
 ;; Expected to be believed.  
 ;; Main goal.  
 ;; Trace of reasoning chain.  
 ;; The node has an alias  
 ;; Reasoning trace results in node e.  
 ;; Points to alias above.  
 ;; Node this IMXP explains.  
 ;; List of all nodes in IMXP.  
 ;; List of all links in IMXP.

---

Figure 32. Successful prediction frame definition

$A \supseteq E$ , then a successful prediction has occurred.<sup>56</sup> Failures occur when  $A \neq E$ . This state exists when either  $A$  and  $E$  are disjoint or there are conflicting assertions within the two nodes. For example  $A$  and  $E$  may be persons, but the concept at node  $E$  contains a slot specifying `gender=male.0`, whereas the concept at  $A$  contains the slot `gender=female.0`. Although successful prediction produces no learning, a representation must exist for it.

Before examining the representation for reasoning failures, it is worth noting that the basic representation of successful prediction can account for many of the process terms in our target ontology (Figure 19 on page 70), not just classes of failure. Figure 33 illustrates successful prediction when the value of the *Cognize* node that produces the expectation,  $E$ , is a memory process. This representation can easily capture the distinctions between an incidental reminding, a deliberate recall, and recognition; that is, the three sub-nodes of *Remember* in Figure 19. The structural differences depend on the nodes  $C$  and  $G$ , and the temporal order of the causal links resulting in nodes  $E$  and  $A$  (see Table 7). If there is no knowledge goal (Ram, 1991; Ram & Cox, 1994; Ram & Hunter, 1992) to retrieve some memory item, only cues in the environment, and if  $E$  is retrieved before  $A$  is produced, then the structure is a reminding. On the other hand, if there is a deliberate attempt to a memory item that is later compared to some feedback,  $A$ , then the structure represents recall. Finally, if  $A$  is presented followed by a memory probe, then the structure represents recognition, whether or not a retrieval goal exists. It is also significant to note that the memory *Elaboration* term of Figure 19 can be represented as a feedback loop from  $E$  to  $C$  such that each new item retrieved adds to the context that enables further memory retrieval.<sup>57</sup> This is represented as a dashed line in Figure 33.

## 4.7 Representing Reasoning Failure to Support Learning

To support learning, a theory should have a level of representation that reflects the structure and content of reasoning failures. Section 3.2, “Types of Reasoning Failure” extends the scope of reasoning failure to include the following forms: contradiction, impasse, false expectation, surprise, and unexpected success. This section provides explicit representations for each of these five types at a level of representation that is sufficient for learning.

---

56. On the other hand, if  $A \subset E$ , then there are more questions remaining on the predicted node  $E$ . If there are unanswered questions, the system will wait for more information before it introspects. Such cases are not represented in the current implementation, although there are cases in which one would want to reason about partial computations. See also the brief discussion in Section 3.2.1.4.

57. This characterization is admittedly simplified since cue elaboration incorporates top-down inferential processes as well as bottom-up additions to memory cues.

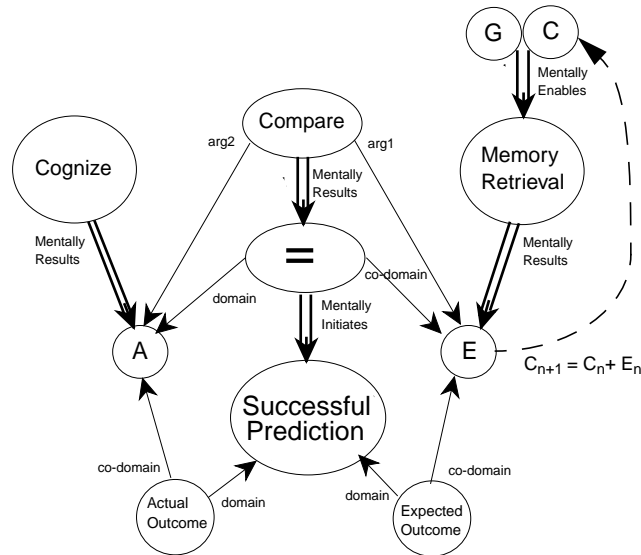


Figure 33. Meta-XP representation of several remembering events

A=actual; E=expected; G=goal; C=context or cues

Table 7: Structural differences between remembering events

Memory Term	Structural Features	Description
Reminding	Has only Cues; E before A	Incidental; No Knowledge Goal
Recall	Cues and Goal; E before A	Deliberate; Has Knowledge Goal
Recognition	May or may not have Goal; A before E	Borderline between 2 above; Has judgement

### 4.7.1 Contradiction

Figure 34 illustrates the representation for a contradiction failure. Some goal, G, and context or cues, C, enables some cognitive process to produce an expected outcome, E. A subsequent cognitive mechanism produces an actual outcome, A, which when compared to E, fails to meet the expectation. Realizing this inequality of actual outcome with expected outcome initiates the knowledge of contradiction.

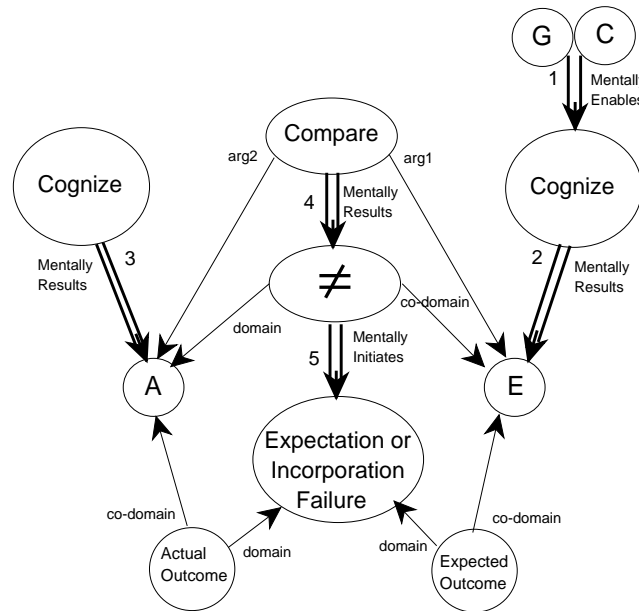


Figure 34. Meta-XP representation of contradiction  
A=actual; E=expected; G=goal; C=context or cues

If the right most Cognize node is an inferential process, then the failure is labeled *Expectation Failure* and the node C represents the context; whereas, if the process was a memory function, the contradiction is labeled *Incorporation Failure* and C represents memory cues. The latter case occurs when an input concept does not meet a conceptual category during understanding. Both inferential expectation failure and incorporation failure are errors of commission. Some explicit expectation was violated by later processing or input.

### 4.7.2 Impasse

Figure 35, “Meta-XP representation of impasse,” represents a class of omission failures that include forgetting as discussed earlier. Some goal, G, and context or cues, C, enables a cognitive process to attempt production of an expected outcome, E. Because the

expectation, E, was not generated, it cannot be compared to an actual outcome, A, produced by a subsequent cognitive mechanism. Realizing that E is not in the set of beliefs with respect to the foreground knowledge of the system (i.e., was not brought into or created within working memory) initiates the knowledge of failure.

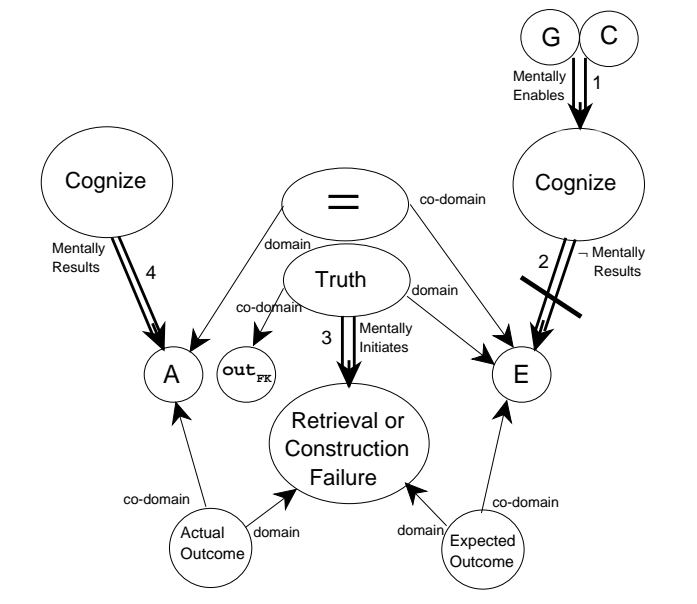


Figure 35. Meta-XP representation of impasse

A=actual; E=expected; G=goal; C=context or cues

If the right-most *Cognize* term is a memory retrieval process, then the Meta-XP indeed represents forgetting,<sup>58</sup> and the structure is labeled *Retrieval Failure*. The impasse is a memory process that fails to retrieve anything. If the node is an inferential process, however, then the impasse failure is equivalent to the failures as recognized by Soar (a blocked attempt to generate the solution to a goal), and the structure is labeled *Construction Failure*. A construction failure occurs when no plan or solution is constructed by the inference process.

### 4.7.3 False Expectation

As seen in Figure 36, the representation of false expectation anticipates an actual event ( $A_1$ ) which never occurs or cannot be calculated. Instead, another event ( $A_2$ ) causes the

58. Compare Figure 35, “Meta-XP representation of impasse,” with Figure 31, “Meta-XP representation of successful prediction,” to see why Forgetting  $\neq$   $\neg$ Remember.



reasoner to realize the error through hindsight. It is not always evident what this second event may be, however. Sometimes it is a very subtle event associated with just the passage of time, so there is no claim here that the second event is a conscious one. In this sequence, the reasoner realizes that the anticipated event is out of the set of beliefs with respect to the FK, and will remain so.

Despite the fact that false expectation and surprise are not closely related in the table of failure types (Table 4, “Final table for reasoning model,” on page 50), they are quite related in structure. As will be seen in the subsequent subsection, they both share the incorrectly anticipated `Successful Prediction` node and also the node labeled `Belated Prediction`.

#### 4.7.4 Surprise

Figure 37, “Meta-XP representation of surprise,” represents a class of failures rarely treated in any AI system. A surprise occurs when a hindsight process reveals that some expectation was never generated. The explanation is that there was never a goal, G2, to create the expectation, either through remembering or inferring. Some earlier process with goal, G1, failed to generate the subsequent goal. When the node A is generated, however, the system realizes that it is missing. This error, by definition, is a missing expectation discovered after the fact. Again, note the similarity between the representations for surprise and false expectation.

#### 4.7.5 Unexpected Success

Finally, Figure 38, “Meta-XP representation of unexpected success,” contains a Meta-XP representation of an unexpected success, a failure similar to contradiction. However, instead of E being violated by A, the expectation is that the violation will occur, yet does not. That is, the agent expects not to be able to perform some computation (e.g., create a solution to a given problem), yet succeeds nonetheless. In such cases the right-most `Cognize` term will be some inferential process. If this process is a memory term instead, the failure represents the case of an agent that does not expect to be able to remember some fact or event when necessary, yet when the time comes, it does nonetheless.

### 4.8 Summary and Discussion

The few examples presented in this chapter demonstrate both the usefulness and complexity of representing mental events and states. The chapter began by describing a target ontology of mental terms that would provide a useful vocabulary for systems that reason about the mental domain. The remainder of the chapter concentrated on composing a rep-



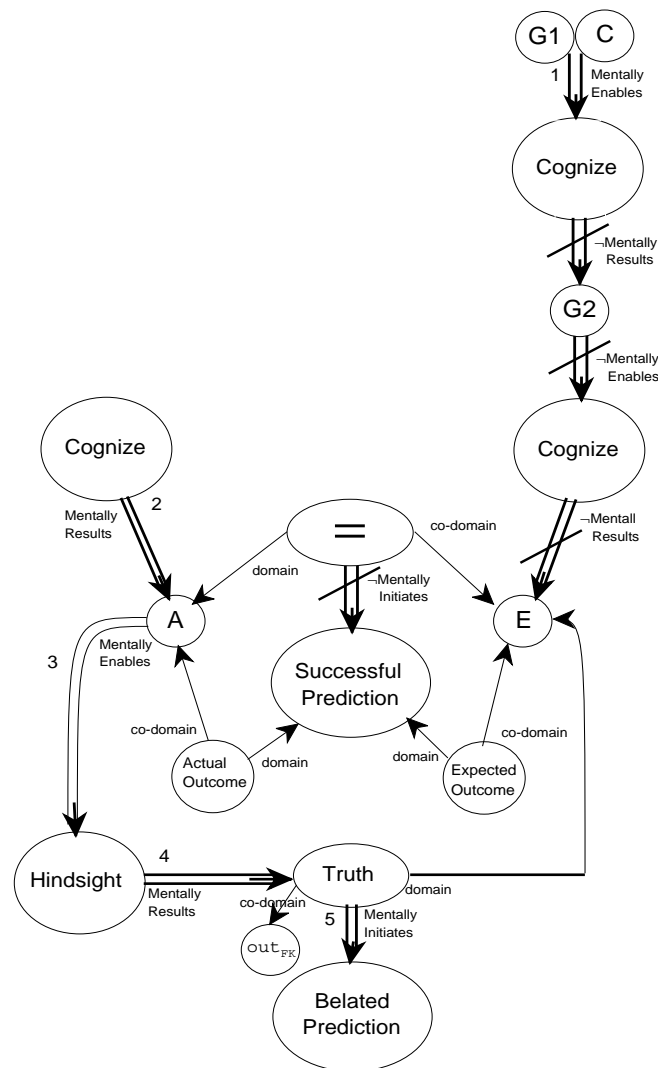


Figure 37. Meta-XP representation of surprise

A=actual; E=expected; G1,G2=goals; C=context or cues

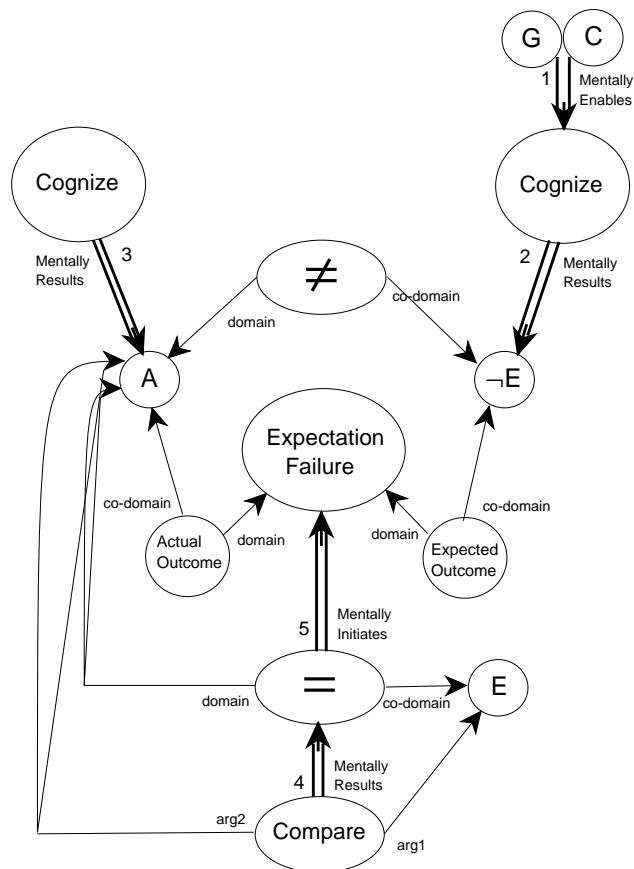


Figure 38. Meta-XP representation of unexpected success

A=actual; E=expected; G=goal; C=context or cues

representation for just those terms that pertain to the five failure symptoms derived in the previous chapter. If a system is to learn from its reasoning failures effectively, it needs to represent the kind of mental symptoms and faults it is likely to encounter so that these can be reasoned about explicitly. Only enough representational detail must be provided so that the system can explain its own failures and thereby learn from them. That is, the representations must have causal and relational components that identify those factors that explain how and why a failure occurred. A knowledge structure called a trace meta-explanation pattern is used to provide a record of system reasoning. It explains how the failures occur. An Introspective Meta-Explanation Pattern represents an abstract causal pattern of failure that explains why the reasoning embodied in a trace fails.

Despite the difficulty of formulating a complete representation of mental events, the effort promises to aid a system when reasoning about itself or other agents, especially when trying to explain why its own or another's reasoning goes astray. Furthermore, even though the CD representation of mental terms leaves much detail unrepresented, the original goal of Schank et al. (1972) to represent the mental domain is a fruitful one. If future research can more fully specify a representational vocabulary for the ontological items illustrated in Figure 19, these domain independent terms can help many different intelligent systems reason in complex situations where errors occur.

Although many of the details of this chapter may be overly simplified, the formalism remains an improvement over many of the representational systems proposed in the past (e.g., logic and CD theory) with respect to representing mental states and mechanisms. Especially considering the emphasis by Schank on expectation as a driving force in text comprehension and problem solving (a point made explicitly as early as Schank, 1972, and, to some extent, in Schank & Tesler, 1969), the CD representation for the concept of “expectation” is not sufficient to express its central role in cognition. For example, the CD representation for “John expects Bill to become a doctor” (Schank et al., 1972, p. 29) is shown in Figure 39. Very little information is provided in this structure, and few inferences may be obtained from it or learning performed from it.



Figure 39. CD representation of expectation

f=future tense; MLOC=Mental Location; LTM=Long

The following chapters of Part Three will introduce the process theory of introspection and learning. Chapter V first presents a model of understanding and then a model of learning. Chapter VII provides the algorithms that underlie these models and that manipulate the current chapter's representations when learning. Additional examples from the Meta-AQUA system will illustrate the utility of Meta-XP representations.

*Part Three*

***A PROCESS THEORY OF LEARNING AND INTROSPECTION***



## CHAPTER V

### A PROCESS THEORY OF UNDERSTANDING AND LEARNING

*Learning without creativity is like a butterfly without wings.*

—Anonymous (from a fortune cookie following a Korean meal).

In most cognitive science theories, problem solving, understanding and learning are distinct processes that have few family resemblances. They each play an integral role in a cognitive milieu, but are, for the most part, studied independently with little regard to one another. But as we have emphasized, the relationship between reasoning (either problem solving or comprehension) and learning is intimate because an introspective learning component must be able to explain and understand failures in the reasoning component, if learning is to remain effective (i.e., if multiple learning methods woven into its learning-strategy are not to interact negatively). Here we develop an interrelated theory of these cognitive functions and show the close relationships between them by comparison and by contrast.

Having examined the content theory of introspective multistrategy learning during Part Two, this chapter presents a process theory of both understanding and learning. Section 5.1 reviews the major suppositions of the theory presented in Parts One and Two. These assumptions also support the models developed in this chapter. Section 5.2 outlines a generalized process theory for multistrategy reasoning that applies to both problem-solving and comprehension tasks. Section 5.3 refines the process theory specifically to comprehension tasks and then specializes it further to account for the task of story understanding. Section 5.4 develops a process model of learning that parallels the model of understanding. Section 5.5 then compares the model of understanding from Section 5.3 with the learning model of Section 5.4.

#### 5.1 Theoretical Assumptions

The results, conclusions and the very structure of this theory depends on the broad assumptions enumerated in Figure 40. Although several sections of this document have

already discussed these assumptions, this section briefly reiterates them to provide a review. These assumptions allow us to be specific as to the kinds of models and the details associated with them that will be presented in this chapter.

- Reasoning is goal-directed processing of input given some background knowledge.
- A multistrategy approach is appropriate for both reasoning and learning.
- Knowledge is memory-based.
- Learning is failure-driven.

---

Figure 40. Assumptions

First and foremost, we assume that cognition is essentially goal-directed processing of a given input using the reasoner's knowledge (see the discussion in Section 3.3). In pursuit of such goals a cognitive system produces expectations of the future. Our focus is therefore on the deliberative and top-down components of thought, rather than on the data-driven or situation-specific factors.

Second, all reasoning can be cast in a multistrategy framework. For the purposes of this document, problem-solving, understanding, and learning are all considered to involve the choice of strategies in some sense. Whereas much of this thesis has already argued that learning is multistrategy affair, Section 5.2 will argue that both problem-solving and understanding should be considered to involve an executive control process that determines a reasoning strategy.

A third assumption is that the reasoner's knowledge and past experience is memory-based, and therefore subject to storage and retrieval constraints, particularly the indexing problem. The indexing problem (see the discussion in Section 3.3.1) is the problem of choosing cues, or features of an input, to be used as indexes for retrieving from memory the knowledge structures necessary to process an input. Thus, in such memories, knowledge organization is a significant concern for both reasoning and learning functions.<sup>59</sup>

Finally, we assume a failure-driven approach to learning and reasoning (see the introductory comments of Chapter III), which concentrates on contradictions, impasses, false expectations, surprises, and unexpected successes to indicate when attentional resources are appropriate. A *failure* is defined as a computational outcome other than what is



expected or a lack of some outcome (or appropriate expectation).

Given such assumptions, the cognitive tasks of reasoning, understanding, and learning have interesting parallels in the overall theory of introspective multistrategy learning.

## 5.2 Multistrategy Reasoning

In a classic study of human problem-solving, Newell and Simon (1972) outlined a model that humans appear to follow when engaged in reasoning about complex problems. An initial process first translates the perception of the external environment into an internal representation of the problem. Second, the reasoner selects a method such as recognition or heuristic search by which to solve the problem.<sup>60</sup> Third, the method is applied to the problem. Finally, if the problem is not solved, then the reasoner either chooses another method, reformulates the problem, or quits. In their framework, the emphasis is upon representation of the problem and multiple problem-solving methods between which the reasoner must decide. We likewise emphasize multistrategy components that select and construct strategies during reasoning and during learning.

In expert system development, a crucial engineering problem is to match an appropriate inference method to the task domain of concern. The generic-task view of Chandrasekaran (1989), Steels' (1990) study of componential frameworks, and McDermott's (1988) work on role-limiting methods all lend additional support for the multistrategy assumption by arguing that different general methods exist that apply to specific problem-solving tasks. More than one method may apply to a given task or subtask, so strategy selection and composition is unavoidable in problem solving, whether performed by the knowledge engineer or the knowledge-based system itself.<sup>61</sup>

---

59. This third premise suggests that knowledge is composed of two parts. As discussed in Section 4.3 on representing forgetting, the system's background knowledge, or BK, contains more than just the domain theory of the performance task. It represents all long-term knowledge including meta-knowledge, heuristic knowledge, associative knowledge, and knowledge of process. In contrast to the BK, the foreground knowledge, or FK, constitutes the current model of the input that has been constructed, and the memory of the reasoning with which such a model was built.

60. See Simon (1979) for a discussion of alternative strategies given a single problem representation.

61. For specific implementations of the multistrategy approach to problem solving, see Goel, Ali, Donnellan, Garza, & Callantine (1994) for descriptions of ROUTER, Punch (1991) for a discussion of the TIPS system, Reinders & Bredeweg (1992) for the REFLECT system, Oehlmann, Edwards & Sleeman (1995) for the IULIAN system, and Kuokka (1990) for the MAX system.

Meta-level tools have also been developed for expert systems to automate the generation of knowledge acquisition assistants. For instance, the PROTÉGÉ-II knowledge acquisition shell (Puerta, Egar, Tu, & Musen, 1992) combines domain- and task-independent mechanisms to construct appropriate problem-solving methods (e.g., heuristic classification and skeletal-plan refinement) that fit a given problem domain. Therefore, rather than one basic mechanism, many different processes and algorithms must account for cognition. The open question is how are they combined. Punch, Goel, and Brown (1996) report a robust mechanism for selecting between a fixed set of alternatives called the sponsor-selector mechanism. This control mechanism has been tested in applications involving planning, design and assembly. Hence, the strategy construction problem (which subsumes the selection task) is clearly pertinent to problem solving, as well as learning.

The first paragraph of Chapter I specified the operational definition of learning as “... given some computational performance task specified by the system’s goals, context and some input, if a failure occurs during the task, the problem is to construct a learning strategy with which to repair the faulty components of the system.” In similar terms, an operational definition of a generalized reasoning task can be stated that subsumes both understanding and problem solving.

*Given some input from the world (e.g., preprocessed perceptual input or text from a story), a current context, including contextual goals and BK, if the input is anomalous (or otherwise interesting), choose or construct a reasoning strategy with which to explain the input.*

As with the characterization of learning, the top level of computation concerns the choice of a reasoning strategy, rather than the choice of a solution operator. The outermost control is thus an executive reasoning process at the meta-level. This multi-level reasoning is consistent with the approach of the MOLGEN system (Stefik, 1981), in which a plane of reasoning exists in both the design plane (the reasoning task in MOLGEN’s domain) and the meta-plane (the task of choosing an operator in the design plane). As a result of this division, to choose a reasoning strategy the system should understand and model its own algorithms. Though consistent with Stefik, however, the model here does not presuppose separate planes of computation.

In the formulation presented here, reasoning is a variant of the generate-and-test paradigm, with the enhancement of a front-end identification process to filter anomalous, or otherwise interesting, input (see Figure 41). So, if no unusual input to the system exists, no significant resources will be expended on reasoning. Therefore, in the absence of interesting input, an understander will skim its data; a problem solver will simply act reactively or habitually. In such situations there is no deliberation. With interesting input, however, a reasoner should construct and execute a strategy, thus generating some response that resolves the anomaly that sparked the interest. Subsequently, the result is verified by some

means constructed by the reasoner. If the result is falsified, then the generation process begins anew.

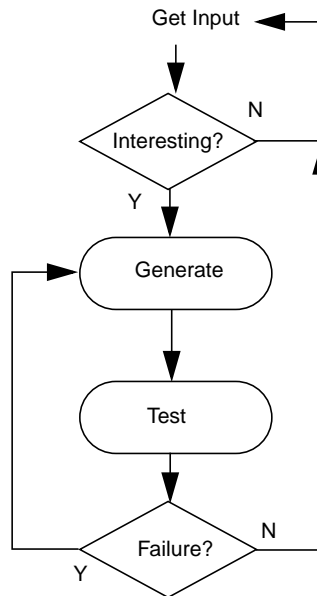


Figure 41. Basic reasoning model

## 5.3 Process Model of Understanding

The research presented here has concentrated on developing the details of first-order reasoning in the form of understanding or comprehension, rather than problem solving. Understanding involves building causal explanations of the input. These explanations provide conceptual coherence by incorporating the current input into pieces of the previous input and by generating expectations about subsequent input. The understander skims a stream of input by instantiating schemas to fit each input item and linking it into the previous concepts of the story, unless the input is anomalous. If an anomalous situation is identified, then the understander must explain the input by elaborating it beyond simple schema instantiation.

### 5.3.1 Three Sub-processes of Understanding

Figure 42 shows three processes in the general understanding task. First, the understander needs to identify anomalous (or otherwise interesting)<sup>62</sup> input; second, it generates a hypothetical explanation to explain the anomaly; and third, it verifies the generated explanation. Both explanation generation and verification involve strategy construction. The understander must construct a method to generate an explanation and to construct a method

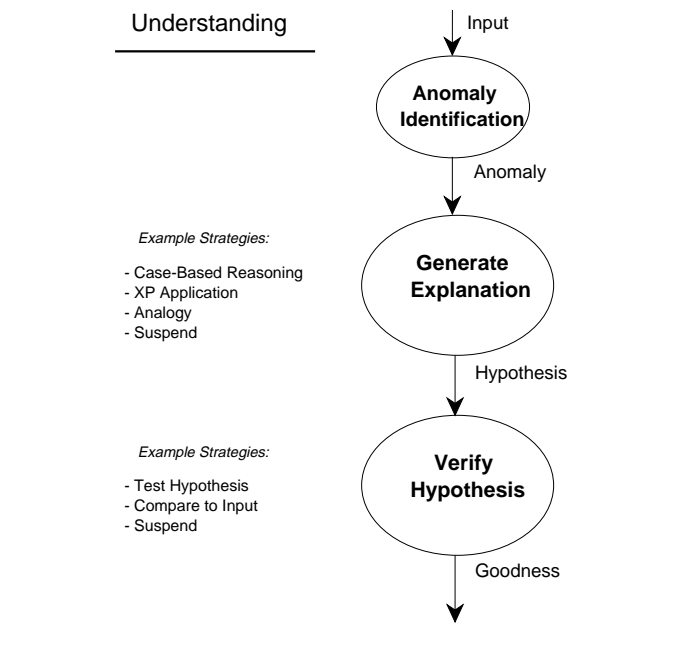


Figure 42. Sub-processes of understanding

to test the veracity of the explanation. A number of possible strategies from which the system may select are shown in Figure 42. With respect to the more generic model shown in Figure 41, the two understanding sub-processes of constructing hypothetical explanations and verifying hypotheses correspond to the generate and test processes, respectively.<sup>63</sup>

With this model, an operational statement of story understanding is as follows:

*Given some input from the story, the system's current foreground knowledge ( $FK_I$ ), including contextual goals and a current representation of the story, and the system's background knowledge ( $BK$ ), if the input is anomalous (or otherwise interesting), choose or construct a strategy with which to*

62. In Meta-AQUA, interesting input is either an anomalous conceptualization or something pertaining to sex, violence, or loud noises. In addition, anything concerning a concept about which something has been learned recently will be categorized as interesting. For a more detailed set of interestingness heuristics, see Ram (1990b).

63. This is not unlike Klahr and Dunbar's (1988) model of scientific discovery, where there is a hypothesis generation phase followed by hypothesis verification and evidence testing phases. The major difference, though, is that IML theory assumes no explicit exploration of a hypothesis space via search. Instead a simple, indexed memory provides suggestions that constitute hypotheses.

*explain the input, else incorporate the input into  $FK_1$ . Output a new story representation ( $FK_2$ ), including a representation of the reasoning that produced it, that has no anomaly and is coherent with respect to the BK.*

### 5.3.2 Understanding Elvis' Behavior

Story-understanding is the processing task chosen to test our theory of introspection and learning. In particular, this research develops an explicit, if somewhat simplified, model of the processing performed by an implementation of question-driven story understanding. The model of understanding used in this thesis is a modification of the reasoning method used by the AQUA story understanding system (Ram, 1991, 1993, 1994). The implementation is in a program called Meta-AQUA.

As an example, Meta-AQUA might process a story about a polite, Memphis musician named Elvis boarding with a young, Southern family.<sup>64</sup> While processing the story, Meta-AQUA constructs a model of the characters and the actions involved in the story. When the story reveals that Elvis occasionally smokes marijuana in the house, endangering his safety and freedom, as well as that of the family's with which he lives, the system detects an anomaly that must be explained to fully understand the story. The event is anomalous because the model of Elvis constructed before the point of his taking drugs was one of a law-abiding citizen. A conflict occurs as a result of trying to unify the picture of Elvis as a typical, adult male (assumed to be happy) with the picture of him as an individual likely to commit a crime (thus, apt to be desperate).

To explain the incongruity, the system must understand the anomaly. Meta-AQUA accomplishes this by consulting a decision model (Ram, 1990a) that describes the planning process an agent performs when considering a choice of actions in the world. The objective of the analysis is to refine the nature of the anomaly and to identify the parts of the story that bear on the anomaly, so as to more clearly ascertain what needs to be explained to resolve the anomaly. An analysis of the story yields the facts that Elvis is not desperate, yet at the same time he performs an act that threatens the loss of his liberty. This situation is certainly anomalous because the decision model asserts that people value the goal of preserving their own freedom above most other goals they possess, other than the goal preserving their lives. A goal competition (Wilensky, 1983) therefore exists that Meta-AQUA must explain.

---

64. Although only outlined in general descriptions here, this scenario is from a 31-sentence story. For a complete program listing of the Meta-AQUA output from this story, see Appendix B. The example will also be dealt with in more detail by Chapter VIII.

Following this analysis, Meta-AQUA poses a series of questions about the anomaly and the context of the story surrounding the anomaly. In this case, the system asks what would cause a man to carry out an action he knew could result in his own arrest. If this question can be answered, then the anomaly would likely be resolved, and the story would be considered understood.

To explain events in a story, Meta-AQUA can generate two types of explanations. *Physical explanations* give a causal account of events according to a model of the way things work in the world, whereas *volitional explanations* give a causal account of why people perform the acts they do in the world. The former class links physical events (such as the burning of flammable materials) with probable causes (such as the lighting of materials with combustible devices). The latter type of explanation links the actions of agents in a story to their goals and beliefs, thus providing a motivation for story characters. In the Elvis scenario, Meta-AQUA retrieves, instantiates, and adapts a cigarette-smoking explanation, which produces expectations in the story (e.g., that the smoking will relieve a nervous emotional state). It can either look for verification of the explanation by tying it into the story, or it can suspend the explanation until a later point in time. The explanation can be verified when subsequent sentences in the story confirm the hypothesis.

Figure 43 diagrams the process decomposition that produces an understanding of Elvis' situation.<sup>65</sup> First the system performs simple anomaly detection. An anomaly is signaled when either the input conflicts with known facts in the BK, or when the system is otherwise unable to successfully incorporate the representation of the input into the current story model in the FK. An explanation process then attempts to resolve the anomaly by constructing a causal account of the input with respect to both the story and the system's knowledge. The resulting hypothesis is then tested for degree of fit or believability.

### 5.3.3 Question-Driven Understanding

Both the generate and the verify components of understanding have four steps (again see Figure 43). In effect, given some anomalous state the reader encounters, if it is to explain the anomaly and thus understand the story, it must answer the following questions:

- How did the anomaly occur?
- What needs to be explained?
- How can I explain this?

---

65. Figure 43 is a repetition of Figure 14 from page 36.

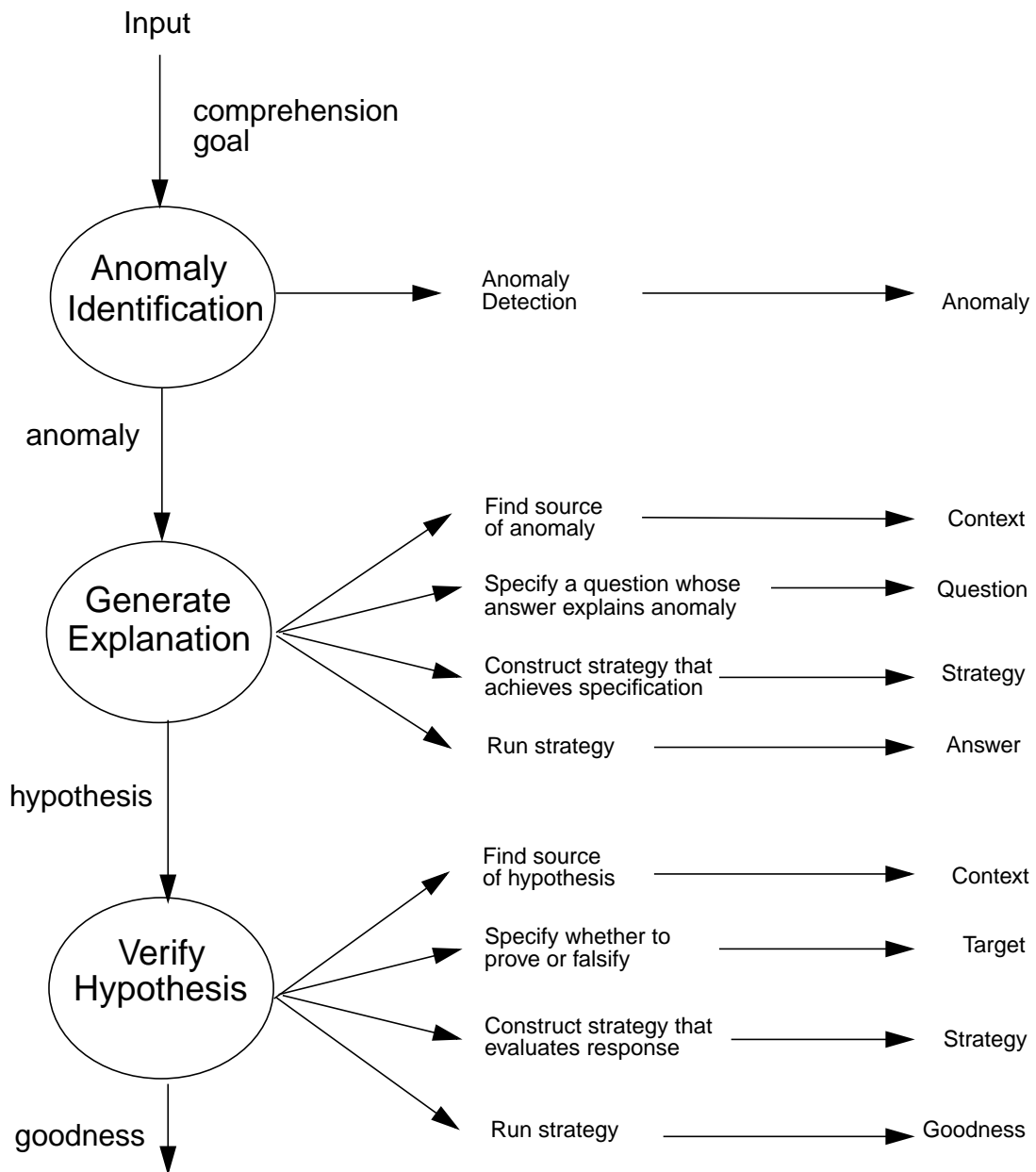


Figure 43. Question-driven understanding

Subsequently it will:

- Resolve the anomaly by generating an explanation.

The initial step is to elaborate the anomaly in order to provide a focus that is relevant to determining what occurred within the story. The reasoner also refines the anomaly in such a way that a specific question can be posed. Since the specification of the explanation process must be more precise than simply “explain the anomaly,” it adds little benefit to simply ask what the reason is for the anomaly. Although it may be clear that some representation for a character like Elvis indicates that he `isa typical-person.0`, that a later representation of him `isa criminal-person.0`, and that the two representations will not unify in the program internals, a better characterization of the anomaly provides specific circumstances (including motivations, states, goals, and beliefs) in terms of both a model of normative decisions and a model of the current story that point to possible locations of the anomaly. Moreover, by providing a story context, a system avoids much search, since the context should contain only the pertinent details known so far. A talented programmer can set up the anomalies that its system knows about in such a way that resolution is all but guaranteed. It is better to have some process that attempts to focus the anomaly so that conditions not envisioned by the programmer can also be addressed.

Given such detail, the function of the next step is to provide a set of questions that represents gaps in the model of the story with respect to the anomaly. Any such question can be viewed as a *knowledge goal* (Cox & Ram, 1995; Ram, 1989, 1991; Ram & Hunter, 1992), since it specifies the knowledge states that, if achieved, would provide coherence to both the story and what the system knows (its BK). Following this specification the system can pick an explanation strategy that will answer these questions (i.e., achieve the knowledge goals). Once a strategy is determined, the program can generate the explanation.

The first step in explanation generation is similar to the blame assignment step in learning, the second is goal specification, and the third is strategy construction. After performing these steps, the reasoner can execute the reasoning method. Like our learning model (see Section 5.4) that offers no new learning algorithms *per se* and instead presents a method of choosing between and combining a number of extant strategies, the reasoning model also concentrates on the strategy choices and combinations. Depending upon the given situation, a system may choose from case-based reasoning, analogy, explanation application, or any number of reasoning strategies for generation (see Figure 43). To perform a test of the resulting hypothesis, a reasoner may devise an experiment, ask someone, or simply wait, in the hope that the answer will be provided by future input.

To verify the hypothesized explanation, the verification process makes a similar four-step analysis. The first step, however, that of finding the source of the hypothesis, is known to follow in sequence from the generation process.<sup>66</sup> Step two is to determine whether to



attempt to prove or disprove the hypothesis. Given a target approach, the system then needs to choose an algorithm best suited to achieving the goal. Once the algorithms have been selected and ordered, the hypothesis can then be evaluated.

Given such a model for the performance task, traces of system performance can be specified and recorded in declarative structures. As described in Section 4.4.1 (starting on page 81), a TMXP contains a decide-compute node (D-C-NODE) for each of the sub-processes of an understanding task; that is, it records the decision and the reasons behind each decision in every step of Figure 43. To understand what parts of the comprehension process are recorded in these knowledge structures, carefully compare Figure 43 (p. 113) with Figure 27 (p. 85). Both the generation and verification processes have four steps each of which correspond to a process field in a D-C-NODE. The four fields are input analysis, goal specification, strategy decision, and strategy execution. For each field, the record stores both the enabling conditions and the resulting state. For the first three fields, the D-C-NODE records the decision basis, and for the last field, it records the side-effects of the process.

If a failure occurs (as detected by the algorithm to be presented on page 121), the system suspends the understanding performance-task and invokes the learning task. When this happens, the trace of the reasoning along with a characterization of the failure (as determined by the failure detection algorithm) is passed to the learning process for introspective explanation. When learning abates, the system resumes the performance task.

## 5.4 Process Model of Learning

In contrast to the first-order performance task, a similar model of the second-order learning task completes our theory of introspective multistrategy learning. When a failure occurs, learning processes inspect the traces of performance in order to explain the failure and decide what to learn. Subsequently, a learning strategy can be assembled and executed. Here we functionally justify such a process model, place it in the context of multistrategy systems, and overview the IML algorithm to be further refined in the next two chapters.

Simon (1983) defines learning as “changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively the next time” (p. 28). Thus, some performance task exists that receives an input and acts upon it given its knowledge dealing with that class of data. A measure of this performance is then passed to a learning task, whereupon it makes

---

66. Yet, in instances where a hypothesis is not self-generated, but provided to the reasoner as input, step one would indeed require significant computation.

changes to the knowledge used by the performance system, depending on the success or failure of the performance. This general view of learning is diagrammed in Figure 44.

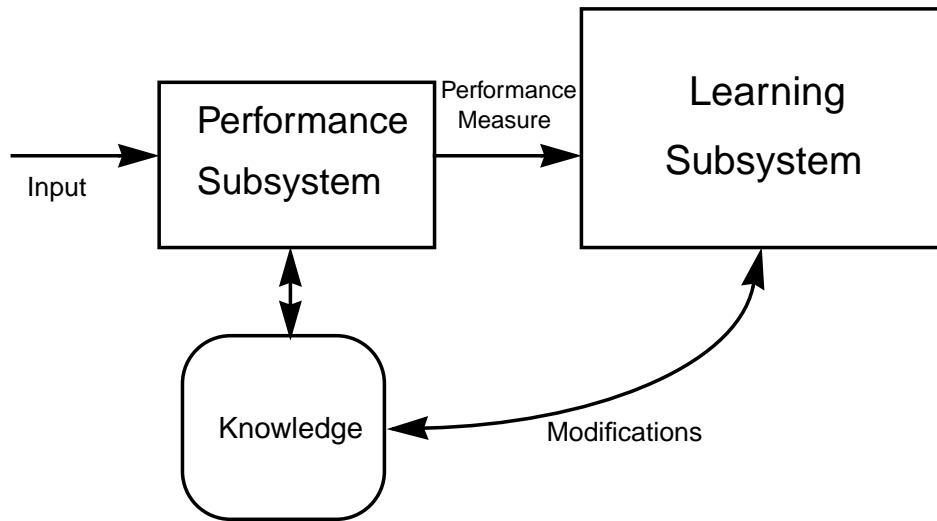


Figure 44. Traditional model of learning

For instance, students often learn to program computers in LISP, first knowing another language such as Pascal. But as LISP novices, the code that results from their problem solving is usually overly-extenuated, inefficient, buggy, and written in an imperative style with loops and block control-structures. As students learn to debug their programs better and acquire mastery of more LISP functions, the code becomes much more compact, efficient, bug-free, and written recursively within a functional programming style. The difference in performance is due to a change in the knowledge and skills used by the programmer both to understand and solve problems and to implement the resulting solutions. These conceptual changes come about from a removal of rigid, Pascal-like coding habits, an acquisition of new LISP techniques, and a reorganization of the applicability conditions for much of the knowledge relevant to the task of computer programming.

In contrast to Simon's definition, the Inferential Learning Theory of Michalski (1991, 1994)<sup>67</sup> defines a learning task as consisting of three components: some input (information), the BK, and a learning goal. Even though this description does not explicitly refer to

67. See also Michalski & Ram (1995) for a more detailed inspection of the relation between views presented here and those of Michalski.

the performance of a reasoning system, and so differs from IML theory, the concept of a learning goal is central to both Michalski's model and the model of learning presented here. The learning goal determines the relevant pieces of the input, the knowledge to be acquired, and the criteria for evaluating the learning. The model of learning presented here is consistent with these constraints, and, as championed by Michalski, concentrates on a multistrategy approach to learning whereby more than one learning strategy can be brought to bear upon a given learning task. Because the multistrategy approach applies equally well to both reasoning (in the form of either problem-solving or understanding) and to learning, this framework is a natural one for integrating the learning and the performance tasks.

### 5.4.1 Multistrategy Learning

Recent attention to multistrategy learning systems is evident from numerous sources in the machine learning literature (e.g., Carbonell, Knoblock & Minton, 1991; Michalski, 1993; Michalski & Tecuci, 1991, 1994) and in the psychological literature (e.g., Anderson, 1983, 1993; Wisniewski & Medin, 1991). Such research constitutes a functional approach that designates the kinds of strategies a learning architecture needs to perform and the conditions for applying each. Multistrategy learning systems are those that integrate several learning algorithms into a unified whole, and thus contrast with single-strategy systems such as Soar (Newell, 1990; Laird et al., 1986; Rosenbloom, Laird, & Newell, 1993) in which all learning is performed by a single learning mechanism. Whereas any learning in Soar reduces to the chunking mechanism, methods as disparate as explanation-based learning, similarity-based learning, deduction, abduction, constructive induction, and analogy can be directly included in the same multistrategy framework. In Soar, such learning strategies must be built up from the chunking mechanism via a production implementation (Steier et al., 1987/1993).<sup>68</sup>

Approaches to multistrategy learning fall into three broad categories, which we call strategy selection models, toolbox models, and cascade models. The common element in all these approaches is the use of multiple learning methods to allow the reasoning system to learn in multiple types of learning situations.

---

68. A more critical evaluation of the single-strategy approach is that learning is actually a melange of several mechanisms of the architecture (Pylyshyn, 1991). Learning can be obtained as a result of goal-driven problem solving (as is with the Soar framework), or by the passive exposure to experience or goal-orientations (for instance, see Barsalou, 1995), or by instruction, by trial and error, by perceptual reorganization or insight, or numerous other mechanisms. The position here, though, is that the question of whether learning is single-strategy or multistrategy is still an open one. This research is simply a start toward the development of a framework that can more vigorously study this question and those like it.

In *strategy selection models*, the reasoner has access to several learning strategies, each represented as a separate algorithm or method. Learning involves an explicit decision stage in which the appropriate learning strategy is identified, followed by a strategy application stage in which the corresponding algorithm is executed. Methods for strategy selection also differ. Pazzani's (1990a; 1994) OCCAM system, for example, tries each learning strategy in a pre-defined order until an applicable one is found; Reich's (1994) BRIDGER system uses a task analysis of the problem-solving task to determine the appropriate learning strategies for each stage of the task; Hunter's (1990a) INVESTIGATOR system represents prerequisites for application of each learning strategy; Cheng's (1995) ISM speedup mechanism manager optimizes the learning behavior of the Theo (Mitchell, Allen, Chalasani, Cheng, Etzioni, Ringuette, & Schlimmer, 1991) problem-solving architecture; and the Meta-AQUA system uses characterizations of reasoning failures to determine what to learn and, in turn, the learning strategies to use when building a learning plan.

*Toolbox models* are similar to strategy selection models in that they too incorporate several learning strategies in a single system. The difference is that these strategies are viewed as tools that can be invoked by the user to perform different types of learning. The tools themselves are available for use by other tools; thus, learning strategies may be organized as co-routines. An example of this approach is Morik's (1994) MOBAL system, in which learning occurs through the cooperation of several learning tools with each other and with the user. Another example of the toolbox class is the PRODIGY (Carbonell et al., 1991; Minton, Carbonell, Etzioni, Knoblock & Kuokka, 1987) system. The system combines explanation-based learning, case-based (analogical) learning, abstraction, experimentation, static analysis, and tutoring. However, the system is designed as a research test-bed for analyzing and comparing different methods, rather than as a system that chooses a learning method itself. Instead, the experimenter chooses a learning module to run against a given problem-solving test suite.<sup>69</sup>

In *cascade models*, two or more learning strategies are cascaded sequentially, with the output of one strategy serving as the input to another. For example, Danyluk's (1994) GEMINI system uses a cascade of explanation-based learning, conceptual clustering, and rule induction strategies, in that order, to combine analytical and empirical learning into a single learning system. Clearly, these categories of models are not exclusive of each other (e.g., a strategy selection system may choose to cascade learning strategies in certain circumstances), but they serve to characterize the major ways in which learning strategies may be integrated.

---

69. The analogy module of PRODIGY is currently being modified to integrate both EBL and abstraction with case-based learning using a cascade model (Veloso & Carbonell, 1994).

Research into multistrategy learning is useful on pragmatic grounds when complex worlds are the domains of learning systems. Such approaches allow for maximal flexibility. Significant interactions are present in multistrategy systems, however, that are not apparent in isolated systems. For example, if two algorithms modify the domain knowledge of the system, and a dependency exists between the two, such that one strategy modifies a part of the domain knowledge that the second one uses, then an implied sequencing must be enforced; that is, the first strategy must be applied before the second. Such dependencies do not exist in single-strategy systems. Research into multistrategy systems contributes to many such issues that pertain to applied systems.

The general model of learning from Figure 44 can be refined to a multistrategy framework as seen in Figure 45. Some input is processed by a multistrategy performance system in a manner dependent upon its goals, expectations, and the knowledge in its memory. A trace of the processing is passed to the learning subsystem. The learning module then uses different methods from a library or toolbox of standard algorithms to make changes to the knowledge in memory. Usually, the tools selected from the toolbox are chosen by the researcher (e.g., as is with the PRODIGY system. See Veloso & Carbonell, 1994, or Carbonell et al., 1991). The goal of this dissertation is to specify a method by which to automate this choice and combination.

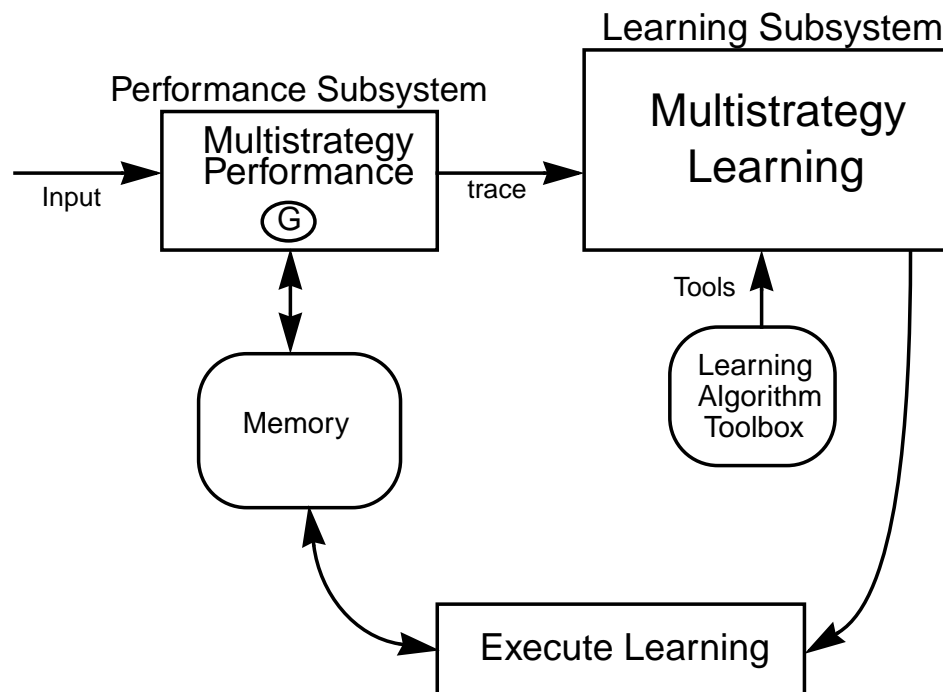


Figure 45. Model of introspective multistrategy learning

### 5.4.2 Process Divisions within the Model of Learning

The performance system records its reasoning during story understanding in a TMXP trace structure as described in Section 4.4.1 (p. 81). These knowledge structures contain representations for each of the reasoning sub-processes: anomaly identification, hypothesis formation, and verification (see Figure 43 on page 113). For each, the structure records the considerations that prompted the process, the bases for making a reasoning strategy decision, and the result of strategy execution. After reasoning completes an inference chain, the reasoning trace is passed to a learning process if a failure is detected.

The learning model itself has three processes. It must monitor the performance system to check for failures, explain and learn from the failure when detected, and, in the ideal model, it should verify that the learning was reasonable. Figure 46 illustrates these three tasks and the information passed between them.

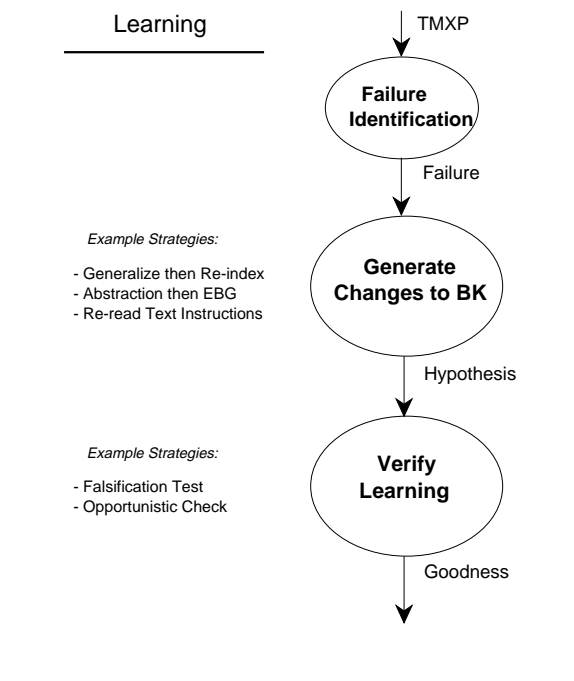


Figure 46. Sub-processes of learning

The first process performs failure detection. As outlined in Section 3.2, five types of failures can occur. Failure detection inputs two structures (an expected outcome, E, and the actual outcome, A) and the trace of the reasoning producing these knowledge structures. The algorithm for this process is shown in Figure 47. The detection process occurs either during the verification phase of the performance task of the system or during the generation

phase after a resumption of a suspended generation goal. This second condition occurs after the performance system previously tried to generate a hypothesis, but could not. The generation phase suspends the goal and new input later provides the answer. See impasse condition in Figure 47. Along with the trace, the process outputs a determination of which of the failures exist (if any) to the next phase.

```

Detect-failure (E, A, trace)
begin
  If (A (outFK) and trace indicates time to event expired)
  or (A (outFK) and impossible (goal (generate, E))) then
    return false expectation
  If E(inFK) then
    if E ≠ A then
      return contradiction
    else if E = A then
      If expected to fail then
        return unexpected success
      else return success
    else if ∃ goal (generate, E) then
      return impasse
    else return surprise
end

```

---

Figure 47. Failure detection algorithm

The second phase concerns the actual determination of the causes of failure and the construction of a learning strategy which is then executed. The strategies from which it may construct a learning plan is dependent upon the IMXP structures in memory. Although this phase will be dealt with in detail by the next section, alternate strategies that may result include combinations of fine-grained knowledge transmutations or more global approaches such as a student's strategy of re-reading instructions when all else fails. The output of the phase is an implicit hypothesis that the learning was correct along with an augmented trace. The changes to the BK from learning are attached to the TMXP and are indexed in memory where the changes occur.

The third phase concerns verification. Although beyond the scope of this thesis and more suitable for future research, verifying the learning could involve either of two strategies. The system could be reminded of a change to the BK (as associated with the TMXP

and described above) at some future time when the changed knowledge is reused. The learning can then be checked as to whether it is effective. Alternatively, the system could actually make a deliberate test of the newly learned knowledge by trying to falsify the information. When either of these processes finish, the verification phase would output an evaluation of the quality of learning.

### 5.4.3 Generating Changes to the BK

Ram and Cox (1994) have argued that three fundamental learning-processes must be performed if learning is to be effective in an open world where many sources of failure exist. The processes are referred to as *blame-assignment* (Birnbaum et al., 1990; Freed, Krulwich, Birnbaum & Collins, 1992; Minsky, 1961/1963; Stroulia, Shankar, Goel & Penberthy, 1992; Weintraub, 1991), *deciding what to learn* (Cox & Ram, 1995; Hunter, 1989, 1990b; Keller, 1986; Krulwich, 1991; Leake & Ram, 1993; Ram, 1991; Ram & Hunter, 1992), and *learning-strategy construction* (Cox & Ram, 1991; Ram & Cox, 1994; Michalski, 1991). In the event of a performance failure, these processes answer the following three questions:<sup>70</sup>

- How did the failure occur?
- What must be learned?
- How can this be learned?

Subsequently the learner will:

- Repair the background knowledge.

To justify our process decomposition that answers these three questions, we advance the following argument: To construct a strategy, a system needs to know what is supposed to be learned; to decide what needs to be learned, it must know the cause of failure; to determine the cause of the failure, it must perform blame assignment; and to perform complete blame assignment in many situations, it must reflect upon its own reasoning. The first subsection to follow elaborates the functional justifications for the process decomposition and the role introspection plays in them, whereas, the second subsection presents an overview of the algorithm that instantiates these processes.

---

70. Note the similarity to the questions on page 112.



### 5.4.3.1 Functional justification for introspection as a component of learning

To properly select an algorithm when constructing a learning strategy, the system must know what it needs to learn; it must have a *learning goal* or target. Imagine that the learning algorithms from which the system chooses are like operators in planning paradigms (Hunter, 1990b; Ram & Hunter, 1992). To select an operator effectively in planning systems, the system must have a goal toward which operators make progress; thus, selection of the actions that constitute steps of a plan is based on the goal of the system. Since operators have results and preconditions, they can be chained such that different operators are chosen on the basis of resultant states that satisfy the preconditions of, and therefore enable, other operators. Thus, they can be chained to produce a series of plan steps that eventually matches the plan goal. Similarly, as plan steps produce changes in the world, learning strategies produce changes in the system's BK. To produce productive changes in the BK, then, the system must have an appropriate learning goal.

Learning goals also provide a focus for learning and thus help to avoid the combinatorial explosion of inferences (Ram & Cox, 1994; Ram & Hunter, 1992). In general, most learning is intractable without some bias. In order to avoid considering all possible input, inferences from this input, and all possible changes to the BK that might improve performance, learning goals give direction to the learner, as do problem-solving goals to a general inference machine. Traditionally, a target concept (i.e., learning goal) is provided to a learning system (e.g., the target concept of a cup in Mitchell et al., 1986). This research enables a system to determine its own target of learning.

Furthermore, to generate the learning goal, the system must know the cause of the failure. Blame assignment (or, conversely, credit assignment) is a well-known problem, going back as far as Minsky (1961/1963), involving the construction of explanations for how and why a failure occurs (or how and why success occurs). Without having knowledge of what caused the system to fail at its reasoning task, it is difficult to know what to learn to avoid subsequent failures in like situations. Perhaps bottom-up reinforcement schedules (e.g., Sutton, 1992) or associative PDP nets (Rumelhart & McClelland, 1986) can help the system learn what to do without it knowing why, but surely no deliberative methods will be able to form a goal to modify the BK in any meaningful way without first analyzing the failure.<sup>71</sup> Explanation is therefore crucial in fully understanding the relation between the current state of the system, its BK, and the current condition of the external world. In this way blame assignment can be viewed as a special form of abduction.

---

71. Pylyshyn (1991) argues the strong position that connectionist nets are essentially statistical pattern matchers. The nets learn directly from the environmental stimulus without the intervention of a reasoning mechanism or interpretation from explicit knowledge.

To perform effective blame assignment, the system must be able to reason about its own reasoning, in addition to reasoning about the world or the results of its own reasoning. Determining the reasons why failure occurs is often not simply a matter of understanding events in the world, or even the plans created. Rather, failures can be attributable to the reasoning process or to the choice of one. Newell and Simon (1972) show that human subjects often make reasoning mistakes because of the wrong choice of reasoning strategy. Given the “Magic Squares” word problem, such that the numbers in some matrix must add up across, up, down, and diagonally, the solution is quite easy using an analogy to tic-tac-toe, but is extremely difficult using means-ends analysis. Unfortunately, most subjects use this latter reasoning and therefore cannot solve the problem. To explain this problem effectively, it is useful to have a mental interpretation of the problem solving process, as well as an explanation that deals with the problem itself.

As another case of the relationship between blame assignment and introspection, consider the stranded motorist example (Cox & Ram, 1992a). If an agent runs out of gas on a vacation, a number of causes could have contributed to the failure. A problem could have occurred with the car’s fuel system (perhaps a hole developed in the gas tank or fuel lines), or a problem could have occurred with the driver’s memory system (perhaps the agent forgot to fill up with gasoline before starting his trip). If the agent is aware of his prior reasoning, including the formulation of a goal to fill up the tank, then when the car rolls to a stop, he should be reminded of the suspended planning goal. The blame is thus associated principally with the mental faculties and the indexes that address the forgotten task, rather than with the physical operation of the car, although there is an unmistakable interaction between the two.<sup>72</sup> One important type of introspection is to realize that the cause of failure was not the plan or solution generated by the reasoner before the trip, but instead was the memory system and the organization of the knowledge that together did not retrieve the suspended goal, given the state of being at or near the gas station. Rather than improve the plan itself, such an analysis can allow a system to improve the organization of the BK by learning better indexes for particular types of suspended goals.

Therefore in many situations, a model of introspection is required to perform blame assignment. Blame assignment is crucial in choosing a learning goal, and the choice of a learning algorithm depends on the learning goal. The following subsections examine the overall introspective multistrategy learning (IML) algorithm and each learning step above in turn.

---

72. Without some naïve knowledge of the physical model of the car, the act of filling the gas tank is meaningless; thus, memory for performing it is mechanical at best.

### 5.4.3.2 Overview of the IML algorithm

Although the following two chapters provide additional details, this section gives an overview of the IML process. Figure 48 sketches the primary algorithm used in introspective multistrategy learning. The system records a trace of the reasoning used in the performance task in a number of TMXPs. Each TMXP is inspected to detect a failure. When the system detects a failure, it invokes learning. During learning, the system constructs a learning strategy with three process steps as described in previous sections. These steps are blame assignment, deciding what to learn, and strategy construction. Subsequently, the system executes the learning strategy to perform the necessary knowledge repairs.

Each of the three subsequent passages presents an operational statement of three main learning processes in the algorithm.

#### **Blame assignment** (step 2a, Figure 48)

*Take as input a trace of the mental and physical events that preceded a reasoning failure; produce as output an explanation of how and why the failure occurred, in terms of the causal factors responsible for the failure.*

Blame assignment is a matter of determining what was responsible for a given failure. Thus, the function of blame assignment is to identify which causal factors (from Table 5, “Detailed taxonomy of causes of reasoning failure,” on page 53) could have led to the reasoning failure as determined from the input (a member of Table 4, “Final table for reasoning model,” on page 50). That is, blame assignment is like troubleshooting; it is a mapping function from failure symptom to failure cause. The purpose is the same whether the troubleshooter is explaining a broken device or itself (Stroulia, 1994).

The input trace describes how results or conclusions were produced by specifying the prior causal chain (both of mental and physical states and events). The learner retrieves an abstract Meta-XP called an IMXP from memory and applies it to the trace in order to produce a specific description of why these conclusions were wrong or inappropriate (the algorithm will be covered in Section 6.2). This instantiation specifies the causal links that would have been responsible for a correct conclusion, and enumerates the difference between the two chains and two conclusions (what was produced and what should have been produced). Finally, the learner outputs the instantiated explanation(s).

0. Perform and Record Reasoning in Trace
1. Failure Detection on Reasoning Trace
2. If Failure Then
 

Learn from Mistake:

  - 2 a. Blame Assignment
    - Compute index as characterization of failure
    - Retrieve Meta-XP
    - Apply Meta-XP to trace of reasoning
    - If Meta-XP application is successful then
      - Check Meta-XP antecedents
      - If one or more nodes not believed then
        - Introspective questioning
        - GOTO step 0
      - Else GOTO step 0
  - 2 b. Create Learning Goals
    - Compute tentative goal priorities
  - 2 c. Choose Learning Algorithm(s)
    - Translate Meta-XP and goals to predicates
    - Pass goals and Meta-XP to planner (Nonlin)
    - Translate resultant plan into frames
  - 2 d. Apply Learning Algorithm(s)
    - Interpret plan as partially ordered network of actions such that primitive actions are algorithm calls
3. Evaluate Learning (not implemented)

---

Figure 48. IML learning algorithm

**Deciding what to learn** (step 2b, Figure 48)

*Take as input a causal explanation of how and why failure occurred; generate as output a set of learning goals which, if achieved, can reduce the likelihood of the failure repeating. Include with the output, both tentative goal-dependencies and priority orderings on the goals.*

The previously instantiated IMXP explanation-pattern assists in this process by specifying points in the reasoning trace most likely to be responsible for the failure. The MetaXP also specifies the suggested type of learning goal to be spawned by this stage. Because these goals are tentative, it may be necessary to retract, decompose, or otherwise adapt the learning goals dynamically during run-time. This stage of learning mediates between the case-based approach of blame assignment and the non-linear planning approach of strategy construction.

The learner includes with the output learning goals both tentative goal-dependencies and priority orderings on the goals. The TMXP is passed as output as well. Section 6.3 on page 143 discusses this phase of the learning task in the context of a working example.

**Learning-strategy construction** (step 2c, Figure 48)

*Take as input a trace of how and why a failure occurred and a set of learning goals along with their dependencies; produce as output an ordered set of learning strategies to apply that will accomplish the goals along with updated dependencies on the set of goals.*

The final learning-strategies are organized as plans to accomplish the learning goals. The plans are sequences of steps representing calls to standard learning algorithms. The plans are created by a Common LISP version of Tate's (1976) Nonlin planner (Ghosh et al., 1992). In order to use the nonlinear planner, the learning module translates the learning goals and the relevant context of the program environment to a predicate representation. In this form, Nonlin assembles a learning plan just as if it were creating a plan to stack a series of labeled blocks. The only difference is that the planner is given a set of learning operators that describe actions that modify the mental world (i.e., the BK) instead of the blocks world.

The learner instantiates the plan, translates it back into a frame representation, and, then executes the learning plans (in step 2d, Figure 48). At the termination of the learning plan execution, control is returned to the performance system. Section 7.2 on page 158 will provide details for the strategy construction phase.

## 5.5 Comparison of Learning and Understanding

Throughout this exposition numerous parallels have been drawn between introspective learning and understanding. Compare Figure 42 with Figure 46, for example, which shows the sub-processes of understanding and learning respectively, as modeled by this research. This chapter has argued, given a multistrategy approach, that a good strategy for both is to identify anomalies, then generate some response to the anomaly, then test the response. The augmented generate-and-test paradigm fits both equally well. Both are concerned with selecting or combining a strategy, rather than applying a particular one. Both models are highly top-down and goal-driven. As many of the arguments advanced in this document show, goals are essential for both focus and direction.

Both the form and the function of the generation phases in learning and understanding are similar (see Figure 49). The structure of both is to take some unusual input (reasoning failure or incongruous story concept), elaborate the input, generate some goal that provides focus for the process, then change some knowledge base to achieve the function of the process. Changes during story understanding take place in the FK, whereas changes during learning take place in the BK.

A number of differences, however, exist between learning and understanding. For example, as understanding can be likened to *recovery*, so too, learning can be likened to *repair*. In the planning literature a number of researchers have made the distinction between recovery and repair (see, for example, Owens, 1991; Hammond, 1989). When a plan fails, the planner must recover from the error so additional progress can be made toward the goal. After recovery, the plan needs to be repaired and stored again in memory, so that the plan failure will not recur.

For example, if an autonomous robot vehicle finds an expected fuel cache missing and thereby runs out of gasoline, it must first recover from the potentially threatening situation by obtaining fuel (example taken from Owens, 1991). Therefore, the explanation of the failure will dictate the means of recovery. If the robot concludes that it cannot find the gasoline because it is lost, then it should recover by obtaining orientation information; whereas, if it explains the fuel's absence because of theft, then the recovery taken will involve turning back or calling for assistance. The repair (to adjust its plans and the information upon which the plan was based) also follows from the explanation of the failure. For instance, if the robot previously considered taking on extra fuel, but did not because it assumed that the fuel cache would be at the proper location and easy to find, then this explanation of its decision would lead the system to modify its knowledge concerning the persistence of fuel caches. This modification would bias it toward conservative decisions in the future, and thus make it less likely to repeat the failure.

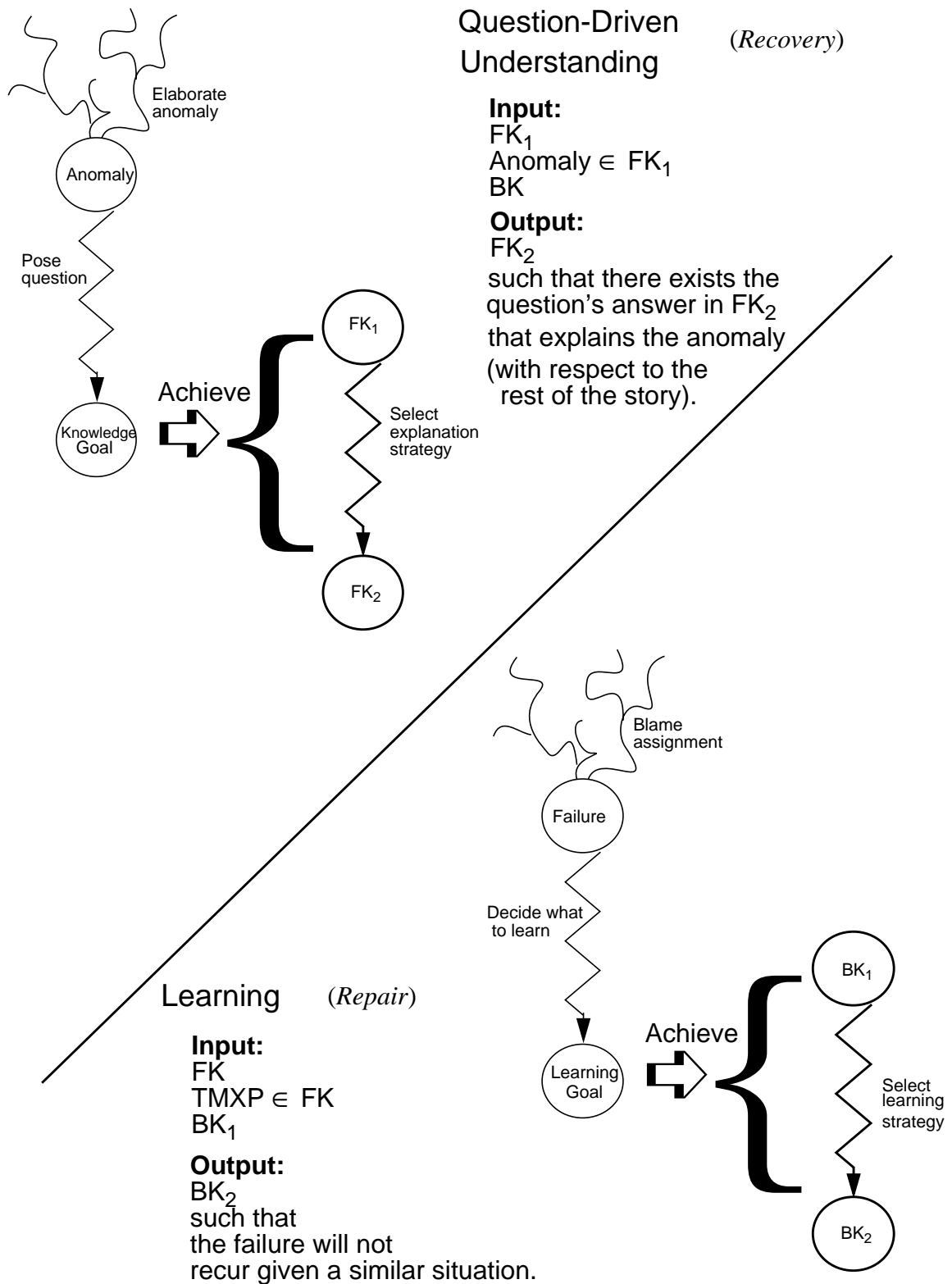


Figure 49. Parallels between learning and understanding

The difference between recovery and repair can be applied to the processes of understanding and learning in an analogous manner. The understanding process requires a recovery phase when it fails. If some explanation does not work, then first there is a need to create a new explanation or somehow to seek one out. Once the correct (or more useful) explanation has been derived, the system needs to learn from the experience by repairing its knowledge, so as not to repeat the failure. Thus, as seen in Figure 49, the understanding process operates on the FK to instill the change that removes the anomaly (thus constituting the recovery); whereas the learning process operates on the BK, producing a repaired knowledge base with which the failure will not be as likely in future similar situations. The recovery is a system's response to anomalous input from the outside world that its knowledge could not adequately understand, whereas the learning is a response to the mental world's inadequacy.

Functional reasons exist for having an explicit input analysis stage in both learning and understanding. Most programs accept cleanly defined problems as input, such that little ambiguity and sharp distinctions concerning what needs to be done exist. In more practical systems, problem elaboration is necessary to clarify what may actually be ill-defined tasks. For example, AI planning tasks are often structured and circumscribed by the programmer or user, not the planner. A planner may be given operational goal specifications from which a state, such as one block being on top of another, may be achieved. The tasks of recognizing that a problem exists for which a plan is required and establishing the goal specifications, however, are not considered a part of the planner's reasoning process. In comprehension tasks such as story understanding, the problems are not usually so well defined. In learning, the problem of recovery is to modify the story representation in such a way that the anomaly is coherent with respect to the rest of the story and the system's BK. This specification is so broad that either the programmer must be very clever, so as to include the specifications implicitly, or the explanation must be somewhat trivial. To narrow the range of behaviors appropriate for recovery, then, is to elaborate the input anomaly, so as to identify what went wrong and why.<sup>73</sup>

## 5.6 Summary

This chapter presented a first-order process theory for understanding and second-order process theory of learning by defining the phases of each, by outlining what steps are used to accomplish such functions, and by arguing why each phase is required by the the-

---

73. Kolodner (1993) also speaks of *situation assessment* (or elaboration) of new input in preparation for case retrieval. The function is the same as input analysis above. In non-trivial systems, a significant part of the problem is to massage the input into a form that is most useful for both processing and retrieval.



ories. IML theory holds that both understanding and learning consists of three phases. The first stage is an input elaboration stage, the second is generation phase, and the final phase is verification. Although each stage is important, this research has concentrated on the generation stage of each. One of the most significant contributions of this research is the specification of the generation stage within learning. This process consists of a step to perform blame assignment, a step to decide what to learn by forming explicit goals to learn, and a final step to construct a learning plan. The plan is then simply executed to accomplish the desired learning.

We have also placed both the reasoning and learning processes within a multistrategy framework. Many methods may exist with which a reasoner can solve a problem, an understander can comprehend an input, and a learner can improve its performance. The choice of a best set of methods and the combination of such methods represent challenging decisions for any intelligent system.



## CHAPTER VI

### CASE-BASED INTROSPECTION

*The best part of human language, properly so called, is derived from reflection on the act of the mind itself.*

—Samuel Taylor Coleridge (1817)

The goals of this chapter are to specify how a system might reflect upon its reasoning so as to understand why its reasoning fails and to describe how it can use this understanding to decide what to learn. The details that address such goals comprise a computational model of introspective explanation and form the first half of our process theory of introspective learning introduced in the previous chapter. Although we do not model all of the attributes one might normally equate with human introspection (such as self-reports on the feelings of pain or perception), we provide a model that begins to account for the human ability to consider one's own, conscious deliberations. By no means does this suggest, however, that Meta-XP structures literally float about within the head, nor does IML theory claim that people have complete and accurate records of past reasoning from which they can make evaluations.<sup>74</sup> Rather, it suggests that certain patterns of reasoning failure are apparent to persons engaged in reflection, and that people can use associated expectations to generate explanations for why they fail at various types of everyday reasoning. In turn, these explanations allow the learner to form specific learning goals.

To make these assertions concrete and to further develop the process theory of learning outlined in the previous chapter, we examine how a multistrategy learning system such as Meta-AQUA introspectively explains its failures and decides what to learn in the context of a story understanding task. Meta-AQUA's task is to create a causally connected conceptual interpretation of an input stream of concepts that represent a story sequence. If the sys-

---

74. Nor does the theory claim that introspection is a computational panacea. See Chapter IX on evaluation of the theory for conditions under which the introspective approach is considered at a disadvantage.

tem fails at this comprehension task, it subsequently learns by (1) using case-based methodologies to analyze and explain the cause of its misunderstanding by retrieving past cases of meta-reasoning; (2) using these past cases to deliberately create a set of learning goals to change its knowledge; and then (3) using nonlinear planning techniques to choose or construct some learning method by which it achieves these goals and hence improve its future performance. Together, these three processes form a hybrid approach combining the use of past experience to generate a set of learning goals and the use of first principles to achieve such goals.<sup>75</sup>

This chapter describes a case-based theory of introspection. In support of this aim, Section 6.1 briefly outlines case-based reasoning from first-order and second order perspectives. The first part of our hybrid model is then illustrated in some detail using the Meta-AQUA example from Section 2.1.1. Section 6.2 explores how blame-assignment (subprocess number 1, above) is performed during introspection, whereas Section 6.3 explains how learning goals are spawned when deciding what to learn (subprocess number 2). Section 6.4 summarizes and concludes by specifying the relation between the Meta-AQUA system and traditional case-based reasoning approaches. The next chapter will continue the Meta-AQUA example begun here to illustrate how learning goals are accomplished using the metaphor of non-linear planning (subprocess number 3).

## 6.1 Case-based reasoning and introspection

As a method for reasoning from concrete past experience, rather than from first principles or abstract rules, *case-based reasoning* (CBR) (Hammond, 1989; Kolodner, 1993; Kolodner & Simpson, 1989; Riesbeck & Schank, 1989; Simpson, 1985) has been presented as a cognitive model of human performance. The approach has been used to model many cognitive activities such as story understanding (Martin, 1990; Ram, 1989), explanation (Kass, Leake, & Owens, 1986; Ram, 1989), planning (Hammond, 1989; Veloso, 1994), design (Hinrichs, 1992), diagnosis (Koton, 1989; Turner, 1989), legal reasoning (Ashley & Rissland, 1988; Bain, 1986), economics reasoning (Pazzani, 1990a), conflict resolution (Simpson, 1985; Sycara, 1987), analogy (Carbonell, 1983, 1986), ethical judgement (Ashley & McLaren, 1995), and troubleshooting (Redmond, 1992). The CBR approach is to retrieve from memory the most similar case, adapt the past case to fit the current situation,

---

75. Conceivably, the assignment of paradigms could have been reversed. A standard planner or problem-solver could have been used to explain the failure and to generate the goals and case-based reasoning could have been used to achieve the goals. Indeed, a fully integrated system might be able to choose which method might be best given dynamically changing conditions (see the discussion in Section 13.1). The reasons to use the particular methods employed here, however, are explained in Section 7.1.2.

then apply the adapted result to the current problem. For example, judges use precedence when issuing sentences for convicted criminals, rather than rules (Bain, 1986). The most similar past legal case is adapted and applied to the current case to derive a sentence. Humans also exhibit this reliance on past examples when learning domains such as mathematics and text-editor skills (Ross, 1989). This chapter proposes that CBR provides a formalism for modeling introspection as well.

As argued in Section 5.4.3.1, to reason effectively about one's own knowledge, goals, and reasoning failure requires an ability to explicitly introspect. A computational model of introspective learning is a second-order theory that contains a formal language for representing first-order processes and that specifies the method of processing and learning from instances represented in this language. The reasoning algorithm used to perform such explanation and learning is similar to the interpretive case-based algorithms used to understand events and processes represented in the original domain.

In general terms, case-based understanding can be specified with the following four steps (Kolodner, 1993):

1. As input, take a representation of some domain episode along with its context and a reasoning goal to provide focus.
2. Based on salient cues in the input, retrieve a prior case to interpret the input.
3. Adapt the old case to fit the current situation.
4. Output the result as the system's understanding of the domain.

When using *case-based introspection*, the system's domain is itself. Therefore, the algorithm is modified as follows:

- 1'. As input, take a representation of some prior failed reasoning (represented with a TMXP) such as an episode of case-based understanding and the goal to explain the failure.
- 2'. Based on salient cues in the input, retrieve a prior case of reflection (represented with an IMXP) to interpret the input.
- 3'. Adapt the old case to fit the current situation.
- 4'. Output the result as the system's self-understanding.

More specifically, such an introspective process can be applied to explicit representations of failed episodes of reasoning and therefore used in the task of reflective blame-assignment. That is, when an understanding system encounters an anomalous input representation, it attempts to explain the anomaly. When explanation fails, a system can use case-based reasoning to retrieve past cases of introspective reasoning that explain the explanation failure. These cases are IMXPs that represent abstract causal patterns of reasoning failure.

## 6.2 Blame Assignment: Explaining reasoning failure

Several computational systems explain reasoning failures. CELIA (Redmond, 1992) is a case-based problem-solving system that learns in the domain of engine diagnosis by observing an expert troubleshooter. In Redmond's theory, four possible failure types exist (one contradiction and three variations of impasse): Incorrect prediction, unexplained connection, unknown goal, and unfamiliar information. For each one, a set of possible root causes exist that could explain the failure. CELIA performs a preliminary mapping from failure (symptom) to cause (fault) by using a discrimination net during initial blame assignment, and uses the resultant general characterization of failure to select a learning (repair) method. Although further blame assignment is performed by the chosen method, the initial causal determination functions as a direct pointer toward relevant repairs. After determining the root cause, CELIA will execute any learning method for which its immediate preconditions are satisfied. Alternatively, Meta-AQUA uses a characterization of failure to retrieve a meta-explanation that can more completely explain the failure; Meta-AQUA's subsequent choice of a learning method is more indirect and mediated by learning goals.

The use of a discrimination net in CELIA explains the failure implicitly. Each node in the net contains a test whose answer determines a directional branching along net links. The terminal leaves of the net contain a characterization of the failure cause, rather than a structural and causal explanation. It is possible that some of the failure explanation could be formally recovered from the tree (depending on the nature of the questions), but this approach is somewhat problematic. In contrast, explicit explanations of failure include prior causal information and relationships that account for the failure and contain a declarative representation that can be analyzed, manipulated, and adapted to more fully meet a given situation (including situations not specifically foreseen by the knowledge engineer). Furthermore, formal explanation structures can be learned as a system encounters new environments or as particular circumstances change.<sup>76</sup> A case-based approach using explanation patterns represents a powerful inference method that welds past experience to the

---

76. The scope of IML theory, however, does not currently cover how new IMXPs are learned. This represents another area for future research.

task of understanding the failure (Schank & Owens, 1987).

When the Meta-AQUA system detects a reasoning failure, the performance module passes a trace of the reasoning to the learning subsystem. At this time, the learner needs to explain why the failure occurred by applying an introspective explanation to the trace. The IMXP is retrieved using the failure characterization as a probe into memory (i.e., the failure type acts as indexing vocabulary). The retrieved meta-explanation is instantiated and bound to the trace of reasoning that preceded the failure. The resulting structure is then checked for applicability. If the IMXP does not apply correctly, then another probe is attempted. An accepted IMXP either provides a set of learning goals that are designed to modify the system's BK or generates additional questions to be posed about the failure. The following section describes the algorithm that is used in this blame-assignment process.

<p>Q3: How to explain a reasoning failure? Ans3: Case-based introspection.</p>
--

### 6.2.1 The Explanation Pattern Application Algorithm

The main control algorithm Meta-AQUA uses for introspective (second-order) explanation is essentially the same as the XP-application control algorithm used in explanatory (first-order) reasoning in both the AQUA (Ram, 1991, 1993, 1994) and SWALE (Kass, Leake, & Owens, 1986; Schank & Leake, 1990) systems. To describe how introspective reasoning works, some background information on XP-application follows.

As described by Section 4.4, the XP knowledge structure is a directed graph that links antecedent conditions to their consequences. The set of sink nodes in the graph is called the PRE-XP-NODES. These nodes represent what must be present in the current situation for the XP to apply. One distinguished node in this set is called the EXPLAINS node. It is bound to the concept that is being explained. For an XP to apply to a given situation, all PRE-XP-NODES must be in the current set of beliefs. If they are not, then the explanation is not appropriate to the situation. If the structure is not rejected, then all source nodes (XP-ASSERTED-NODES) are checked. For each XP-ASSERTED node verified, all adjacent nodes are marked as verified, and the verification marks are propagated toward the EXPLAINS node. If all XP-ASSERTED-NODES can be verified, then the entire explanation is verified. Gaps in the explanation occur when one or more XP-ASSERTED-NODES remain unverified. Each gap results in a question, which provides the system with a focus for reasoning and learning, and limits the inferences pursued by the system.

Given this methodology, the algorithm for explaining and learning from a reasoning failure works much the same way. Step 2 of Figure 48 on page 126 outlines the control algorithm for blame assignment in an introspective multistrategy learner. The step is refined in Figure 50 below.

1. Compute index as characterization of failure
2. Retrieve IMXP
3. Apply IMXP to TMXP trace of reasoning
4. If  $[\forall X \in \text{PRE-XP-NODES} . X(\text{in}_{FK})]$  then
  - If  $[\exists Y \in \text{XP-ASSERTED-NODES} . \neg Y(\text{in}_{FK})]$  then
    - Recursively explain  $\neg Y$
    - If cannot immediately explain  $\neg Y$  then
      - reject IMXP
  - Else If  $[\exists Z \in \text{XP-ASSERTED-NODES} . Z(\text{out}_{FK})]$  then
    - Recursively pose the question “Is Z believable?”

---

Figure 50. Reflective blame assignment

The identification of blame during the learning phase is analogous to the method above used in AQUA or SWALE to explain anomalies in story inputs. Instead of taking as input a conceptual representation of events in the world and outputting an explanation of the anomaly, however, the blame assignment process in Meta-AQUA takes as input a conceptual representation of the reasoning performed in explaining an event in the world and outputs an explanation of the reasoning failure. Just as the XP application algorithm can be applied to events in the world, the reflective blame-assignment algorithm in Figure 50 can be applied to a set of mental events, using Meta-XPs with a single level of recursion.

A characterization of the reasoning failure from the system’s vocabulary of failure terms is used as an index to retrieve an abstract IMXP. Here the characterization serves as a probe into memory, rather than providing a direct explanation. The retrieved IMXP structure is then bound with the trace of the reasoning to produce a parameterized token. The PRE-XP-NODES are then checked to see if they are consistent with the current representation of the reasoning that produced an understanding the story. If they all can be verified then the Meta-XP applies to the situation. If any are rejected, then the explanation is rejected. If any of the nodes are neither confirmed nor rejected, a question is posed on the node. When the question is not answered, the introspection is suspended, the reasoning is indexed in memory and the performance task is resumed. When future opportunities warrant, the system can resume the introspective process.



More specifically, Figure 50 specifies that the IMXP applies if all PRE-XP-NODES are in the set of beliefs with respect to the FK. If this is so, then the system makes sure that no element of the set of XP-ASSERTED-NODES is contradicted by something in the story model or reasoning model within the FK. But if this is so, it tries to recursively explain this sub-anomaly. If the anomalous item cannot be immediately explained, it rejects the explanation and searches for another one.<sup>77</sup>

If no anomalous nodes exist in the XP-ASSERTED-NODES, then the system checks to see if all nodes in the set of XP-ASSERTED-NODES are in the set of beliefs with respect to the FK (i.e., all are believed). If all are believed, it accepts the explanation.

If any nodes are not yet believed, the system poses a question to find out whether or not they can be believed (i.e., to determine if the system can infer the belief). The representation for the recursive question “Is Z believable?” is listed in Figure 51. That is, literally it asserts “The truth value of Z is in the set of beliefs with respect to the foreground knowledge, no?” This question is an introspective question because it is posed on a representation of part of the reasoning, rather than on a representation of a part of the story (see Oehlmann, Sleeman, & Edwards, 1993, for a related model of introspective questioning).

```
(truth
  (domain
    (value Z))
  (co-domain
    (value (inFK)))
  (status
    (value question.0))))
```

---

Figure 51. Representation of the question “Is Z believable?”

Once an IMXP is retrieved and successfully applied to the trace of failed reasoning provided by some TMXP, the system must generate a set of learning goals as previously outlined. This process will be covered in Section 6.3. But first, the following section places the process into context using a Meta-AQUA example from Section 2.1.

---

77. The possibility exists that a wait-and-see strategy might work at this point. Thus, a system could suspend the explanation and index it into memory. The opportunistic approach hopes that the anomaly can be explained later. Although Meta-AQUA does not currently use this strategy, if used, it is probably better to wait until all other explanations are exhausted first.

### 6.2.2 Returning to the Drug Bust Example

As presented by Chapter II, Meta-AQUA was written to test our theory of understanding, introspection, and learning. Given the drug-bust story of Figure 52 (reprinted from Figure 11), the system attempts to understand each sentence by incorporating it into its current story representation, explain any anomalous or interesting features of the story, and learn from any reasoning failures. Numerous incorrect inferences can be made from this story, depending on the knowledge of the reader. Meta-AQUA's background knowledge includes general facts about dogs and sniffing, including the fact that dogs bark when threatened, but it has no knowledge of police dogs. It also knows cases of gun smuggling, but has never seen drug interdiction.

S1:A police dog sniffed at a passenger's luggage in the airport terminal.  
 S2:The dog suddenly began to bark at the luggage.  
 S3:The authorities arrested the passenger, charging him with smuggling drugs.  
 S4:The dog barked because it detected two kilograms of marijuana in the luggage.

---

Figure 52. The drug-bust story (HC1)

As will be recalled, Meta-AQUA had detected an anomaly when the dog barked at the luggage because it had only experienced dogs barking at animate objects (see program output in Figure 53). The program then naively explained the anomaly as the dog's response to a threat from the luggage. When the story provides a better explanation of the barking episode, the system concludes that the reasoning had produced a faulty explanation, and therefore it enters a learning cycle.

The system characterizes the reasoning error as a contradiction caused by the incorrect retrieval of a known explanation ("dogs bark when threatened by objects," erroneously assumed to be applicable), and a missing explanation ("the dog barked because it detected marijuana," the correct explanation in this case). During blame assignment, Meta-AQUA uses this characterization as an index to retrieve an abstract case called a Meta-XP that is applied to a trace of the reasoning that produced the failure. Figure 54 shows the instantiated result in an explanation of its reasoning error. This composite meta-explanation consists of three parts: a Novel-Situation centered around Retrieval Failure, an Erroneous-Association centered around Expectation Failure and an Incorrect-Domain-Knowledge centered around Incorporation Failure.

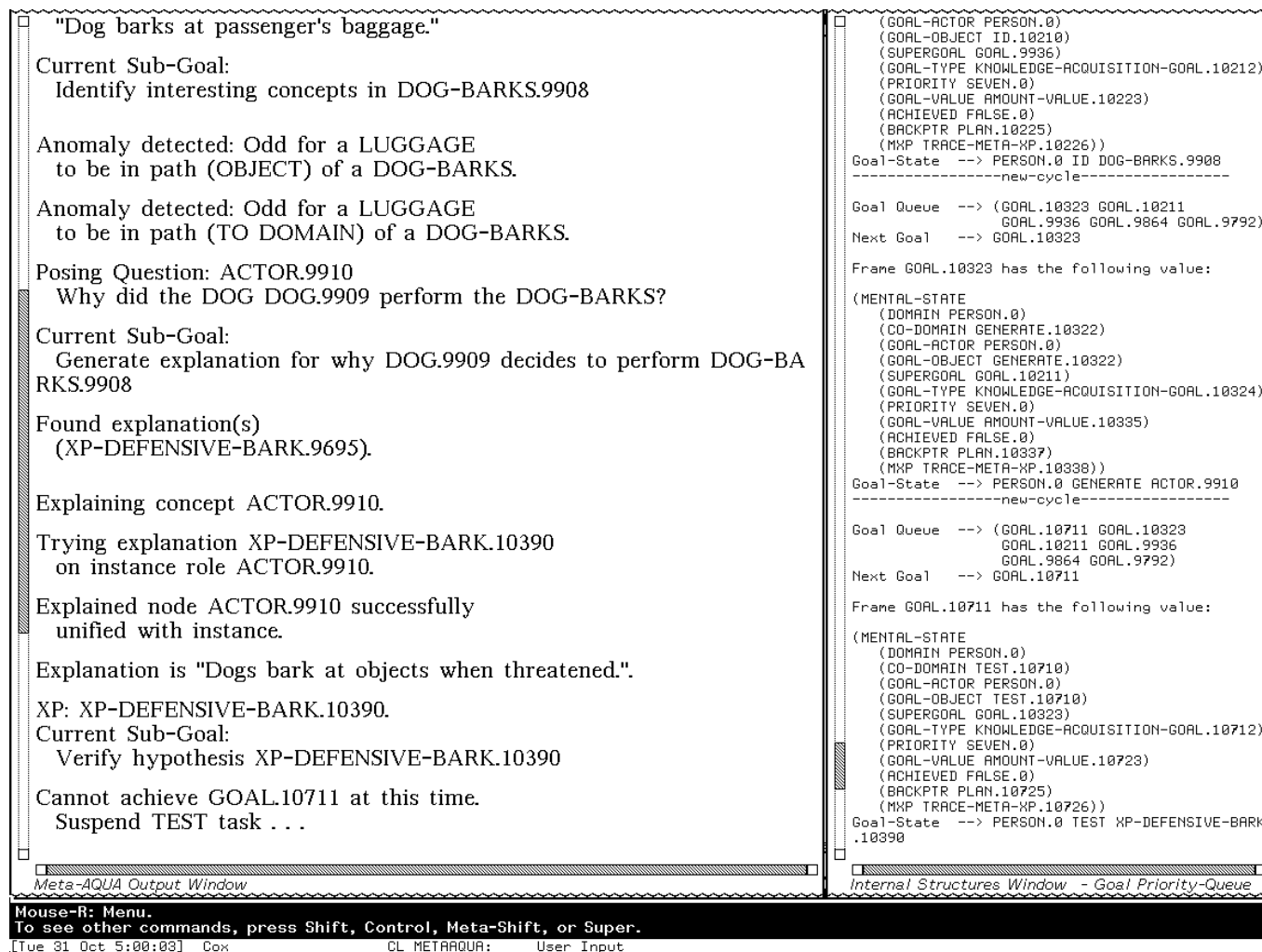


Figure 53. Meta-AQUA output during hypothesis generation of HC1

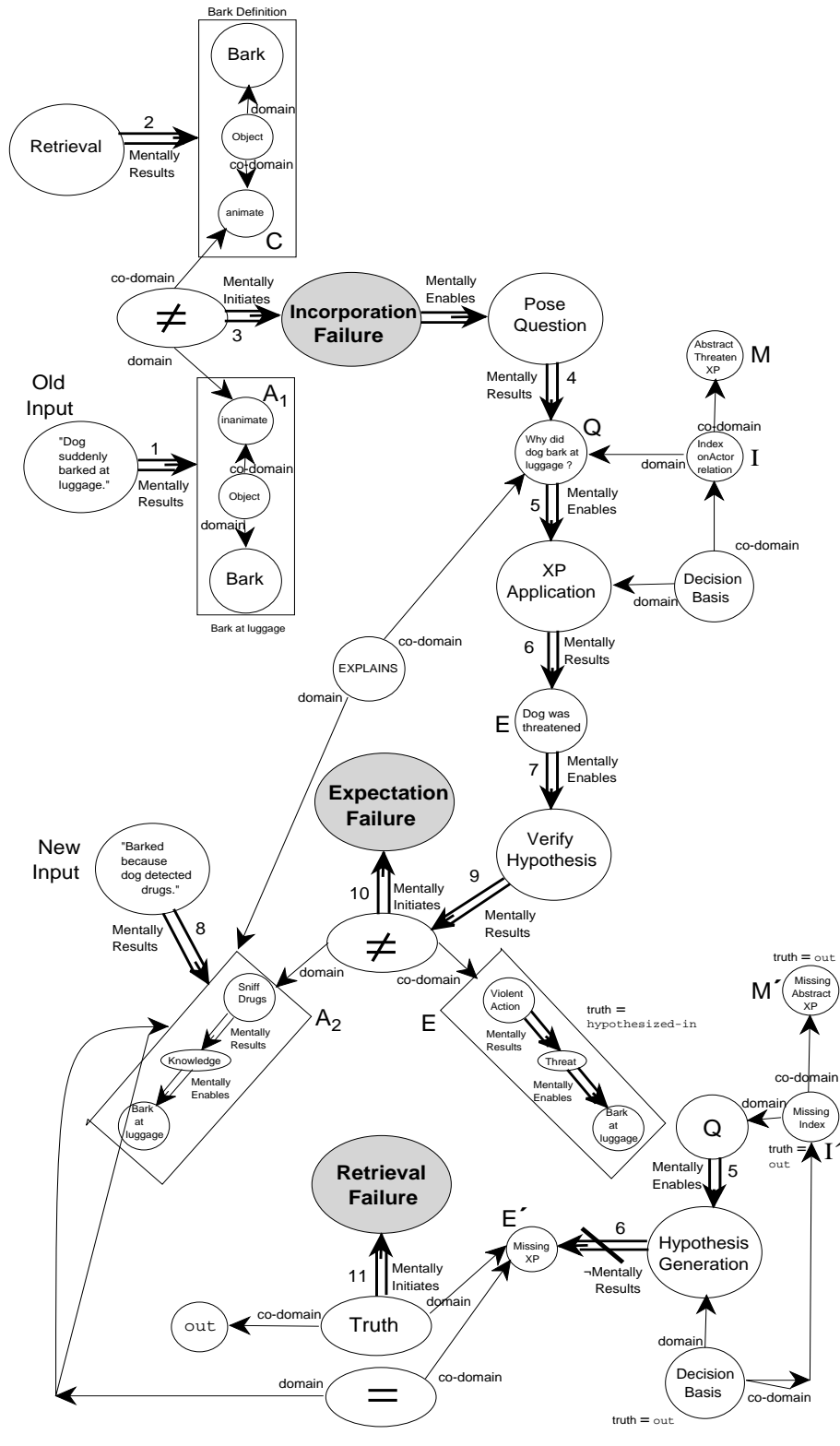


Figure 54. Instantiated IMXP for mis-explanation

The abstract IMXP from which this instantiation originates, IMXP-NOVEL-SITUATION-ALTERNATIVE-REFUTED, captures a common pattern of failure in systems that are learning new concepts. When a concept is being learned, it may be overly specialized. Slight variation on the concept will cause the system to try to explain it, but without experience with the concept, the system may generate an inappropriate explanation. The proper explanation may not be known because the situation is novel.

As seen in Figure 54, the vertical chain of processes starting with the node labeled “Pose Question” represents part of a TMXP. This trace records the decisions preceding the detection of the explanation failure (i.e., that the dog was actually barking because it detected the contraband, not out of defensive instincts). The IMXP structure formally explains the node labeled *Expectation Failure*, although in general, it gives the causal chain of events for much of the reasoning associated with all parts of the error.<sup>78</sup> To check whether or not this explanation applies to the failure, Meta-AQUA checks the truth values of nodes A2, E, and the EXPLAINS node. Because these already exist as known entities in the representations (i.e.,  $in_{FK}$ ), the XP is accepted.

Although the information in Figure 54 appears to be complex, the IMXP simplifies the amount of detail the reasoner must consider during blame assignment by abstracting away much of this information. To show what the system actually considers, Figure 55 represents an overlay that corresponds to Figure 54 (mentally align the shaded nodes, such as the *Incorporation Failure* marked IF, between the two figures to see the simplification Figure 55 provides). The remaining details are not considered in depth during evaluation.

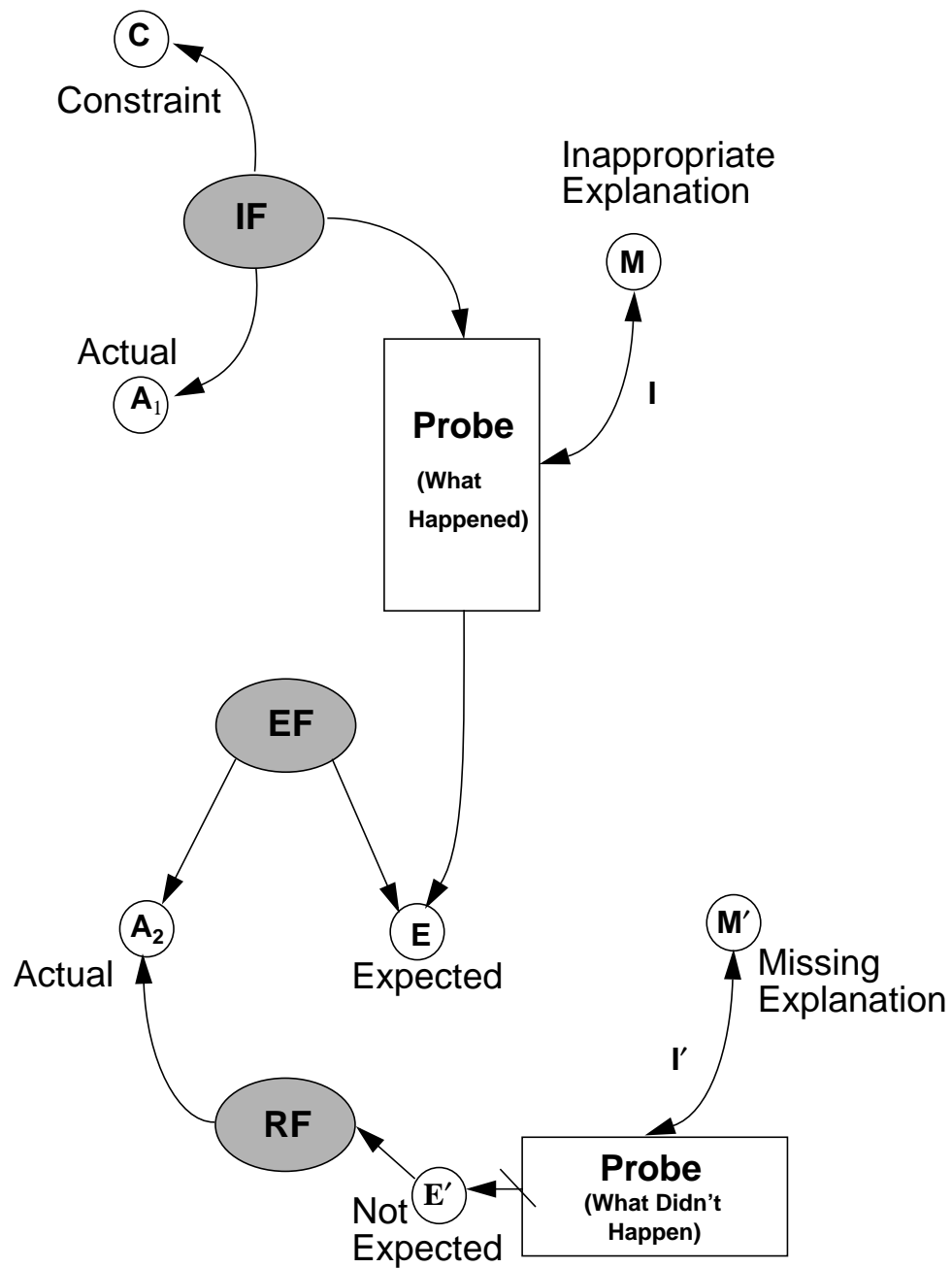
A screen shot of the Meta-AQUA output during the learning associated with this example is shown in Figure 56. The left window in the screen displays processing output, whereas the right side shows internal representations. In this case, the right half of the figure shows the outermost instantiated frame representation of the IMXP depicted as a graph in Figure 54.

## 6.3 Deciding What to Learn: Spawning learning goals

Given a reasoning failure, the task of the learning system is to adjust its knowledge so that such reasoning failures will not recur in similar situations. To perform this adjustment, the learner constructs a learning strategy that is designed to satisfy specific learning goals.

---

78. Note that the node labeled “EXPLAINS” in Figure 54 is the EXPLAINS node for the XP labeled A<sub>2</sub>, not for the IMXP itself.




---

Figure 55. IMXP abstraction of causal factors involved in story HC1

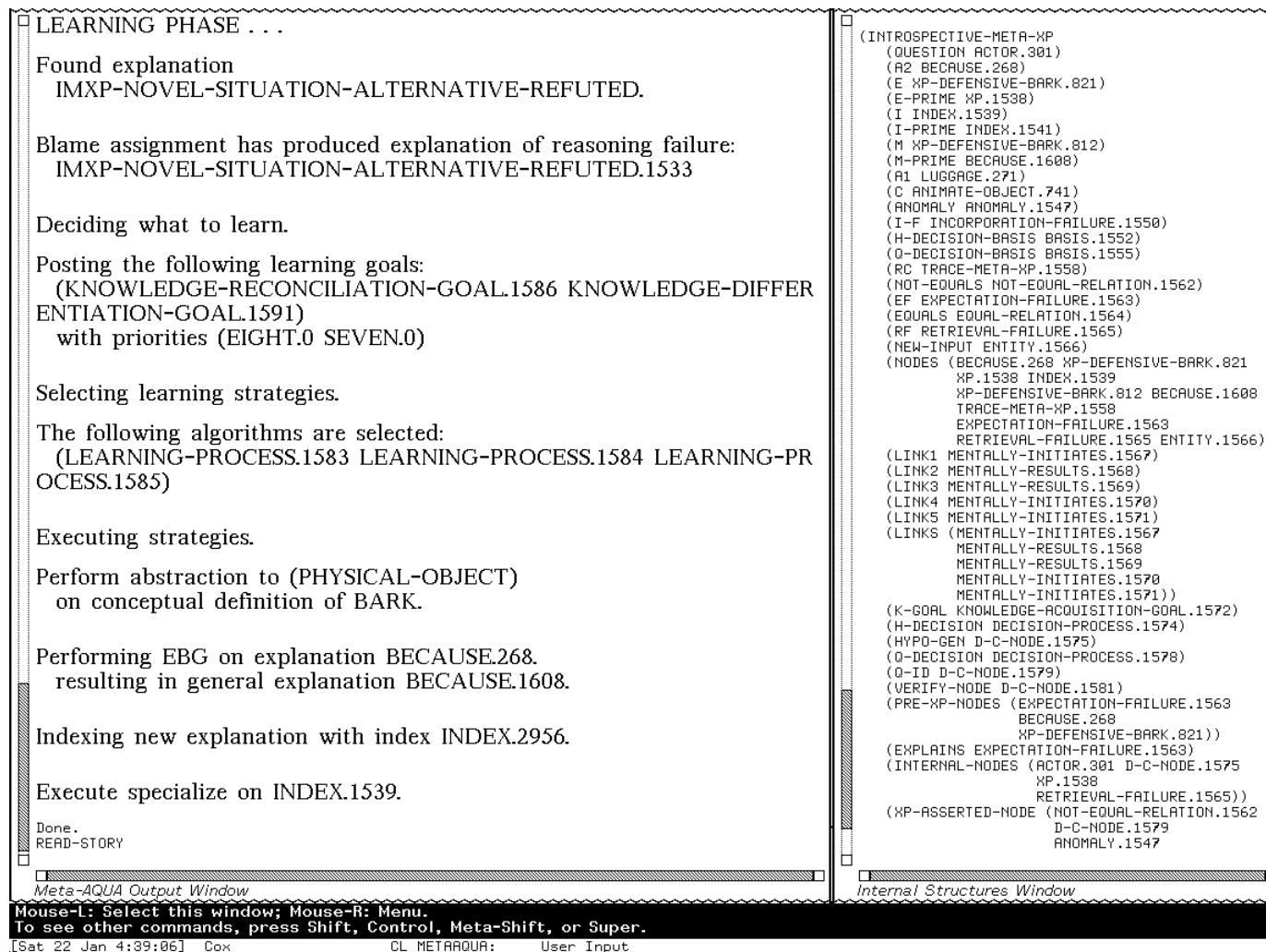


Figure 56. Learning output and frame representation of the Meta-XP used in example story

These goals are created in a decision process that circumscribes what needs to be learned. The decision is made in response to the explanation of failure generated during reflective blame assignment. The overall aim of the system is to turn reasoning failures into opportunities to learn and to improve the system's performance.

### 6.3.1 Explicit Learning Goals

To provide focus for learning, a set of learning goals are spawned from the explanation of the failure. In many theories, goals are an implicit, desired state and have no special representation other than the representation for that state itself. Newell (1982) considers a goal as “a body of knowledge of a state of affairs in the environment” (p. 101), but one especially distinguished from common bodies of knowledge so that an agent will strive to achieve them. In all other structural ways, goals are the same as other bodies of knowledge. Newell & Simon (1972) consider the methods for achieving goals as part of the goal structure itself. In our approach, goals are state descriptions with annotations and a particular structure. The methods for achieving goals and the knowledge associated with the states they seek are considered separate entities.

Schank & Abelson (1977) enumerate a goal taxonomy that contains seven different types of goals. IML theory collapses these seven into three broad distinctions (see Figure 57). An *achievement goal* represents a state an agent wishes to achieve in the world or in the BK. These goals represent the traditional interpretation of goal in that the state is currently not what is desired, and to achieve the goal state some operations will be required to accomplish transformations that result in the goal state. Contrastingly, a *prevention goal* is an undesired state that the reasoner wishes to avoid, and a *maintenance goal* is a state that the reasoner wishes to preserve. Both of these two type of states may be achieved by setting up conditions that affect other agents (such as threats) or by doing nothing at all except monitoring the states and preparing contingency plans.

- Achievement Goal
- Prevention Goal
- Maintenance Goal

---

Figure 57. Broad goal categories

Figure 58 depicts the generic frame definition for a goal; whereas, the end of Appendix D contains instantiated goal examples. A goal is essentially a binary relation from volitional-agent to a desired state (i.e., relation) in either the world or in a body of knowledge. Goals are arranged in hierarchies such that the achievement of subgoals contribute to the achievement of supergoals. From a given goal, pointers exists to both superiors and subordinates in the hierarchy, as well as a direct pointer to the plan for which they are in service.



Also associated with each goal is a type designation, a flag that indicates whether it is successful, a relative priority, and an absolute estimate of usefulness. Most importantly, a pointer exists with each goal to the trace of the reasoning that spawned and pursued the goal. When the goals are suspended and later resumed during opportunistic reasoning, the reasoner can use this pointer to locate the TMXP required to resume the processing associated with the goal (whether the goal is pursuing a state in the world or a change in the BK).

```
(define-frame GOAL
  (isa      (value (mental-state)))
  (domain   (constraint
              (volitional-agent)))
  (co-domain (constraint
                    (relation)))
  (goal-actor (value =domain))
  (goal-object (value =co-domain))
  (goal-type  (constraint
                (goal-type-value)))
  (supergoal  (constraint
                (goal)))
  (subgoals   (value nil.0))
  (priority   (constraint
                (integer-value)))
  (goal-value (constraint
                (amount-value)))
  (achieved   (value
                false.0))
  (backptr    (constraint
                (plan))
               (default
                nil.0))
  (MXP        (constraint
                (trace-meta-xp)))
)
```

;; A goal is a mapping from a volitional-agent  
 ;; (i.e., whose goal is this),  
 ;; to some desired relation in the world or  
 ;; in the mind.  
 ;; Useful synonym.  
 ;; Another useful synonym.  
 ;; Goal's taxonomic tag.  
 ;; E.g., knowledge-acquisition-goal.0.  
 ;; Link to parent goal in goal hierarchy.  
 ;; Will be a list of goals in hierarchy.  
 ;; Current relative goal priority.  
 ;; Crude measure of absolute usefulness.  
 ;; Goals begin as unachieved (false.0),  
 ;; becomes true.0 when satisfied.  
 ;; Back pointer  
 ;; to plan that spawned this goal.  
 ;; If created for another reason,  
 ;; will be nil.  
 ;; Points to the TMXP  
 ;; to which this goal is associated.

Figure 58. Frame definition for goals

Just as standard goals represent what an agent needs in the world, learning goals represent what a system needs to know (Cox & Ram, 1994b; Ram, 1991; 1993; Ram & Hunter, 1992; Ram & Leake, 1995). These goals are spawned when deciding what to learn or when subgoalting on a superordinate learning goal. Learning goals help guide the learning process by suggesting strategies that would allow the system to learn knowledge required to avoid future failures. Learning goals specify the desired structure and content of knowledge, as well as the ways in which knowledge is organized in memory. Learning goals also facilitate opportunistic learning (see Hammond, Converse, Marks, & Seifert, 1993; Ram, 1991; 1993; Ram & Hunter, 1992); that is, if all information necessary for learning is not available at the time it is determined what is needed to be learned (e.g., when a question is posed), then a learning goal can be suspended, indexed in memory, and resumed at a later time when the information becomes available.

Figure 59 lists the types of learning goals used in IML theory. All of these goals are achievement goals because they attempt to achieve some new state in the BK, rather than prevent or maintain some state.<sup>79</sup> Some learning goals seek to add, delete, generalize or

## Unary Goals

- Knowledge Acquisition Goal
- Knowledge Refinement Goal
- Knowledge Expansion Goal
- Knowledge Organization Goal

## N-ary Goals

- Knowledge Differentiation Goal
- Knowledge Reconciliation Goal
- Knowledge Organization Goal

---

Figure 59. A taxonomy of learning goals

specialize a given concept or procedure. Others deal with the ontology of the knowledge, that is, with the kinds of categories that constitute particular concepts. Many learning goals are unary in that they take a single target as argument. For example, a *knowledge acquisition goal* seeks to determine a single piece of missing knowledge, such as the answer to a particular question. A *knowledge refinement goal* seeks a more specialized interpretation for a given concept in memory, whereas a *knowledge expansion goal* seeks a broader interpretation that explores connections with related concepts.

Other learning goals are n-ary in nature because they take two or more arguments. A *knowledge differentiation goal* is a goal to determine a change in a body of knowledge such that two or more items are separated conceptually. In contrast, a *knowledge reconciliation goal* is one that seeks to merge multiple items that were mistakenly considered separate entities.

Both expansion goals and reconciliation goals may include/spawn a *knowledge organization goal* that seeks to reach a particular configuration of the BK. Usually a learner wants to reorganize the existing knowledge so that it is made available to the reasoner at the appropriate time, as well as modify the structure or content of a concept itself. Such reorganization of knowledge affects the conditions under which a particular piece of knowledge is retrieved or the kinds of indexes associated with an item in memory. Because the desire can be either to organize a particular state or to organize two or more states with respect to each other, this goal type may take one or more arguments.

---

79. Of course, learning goals that either prevent the removal of a strongly held belief or maintain some special mental state are conceivable, but outside the scope of this research.

A list of specific learning-goals that need to be spawned are included as part of the representational structure of each composite IMXP. When the IMXP is bound to the trace of the failure, the goals are automatically bound to particular points in the representation of the trace (via unification of variables) that provide the most likely location of failure. As will be seen in the next chapter, these are only starting points; additional subgoals may be spawned to support the original goals from the IMXP. The goals are placed on a priority queue of current goals to be pursued by the next stage of the learning process.

Q2: How to decide what to learn?  
Ans2: Post a learning goal.

### 6.3.2 Learning Goals in the Drug-Bust Example

Faced with the structure of the reasoning error produced by the blame-assignment phase, the learner determines the learning goals for the system. First, since the seemingly anomalous input (marked “Old Input” in Figure 54) has been incorporated into the story and later reinforced by the coherence of the story structure, and since no contradictions occurred as a result of this inference, the learner posts a knowledge reconciliation goal (see G1 in Figure 60). The goal is to adjust the background knowledge so that neither dogs barking at animate objects nor dogs barking at inanimate objects will be considered anomalous by the understander. This learning goal is appropriate because even though one item is an instantiated token (a particular dog barked at a specific inanimate object), while the other is a type definition (concept specifying that dogs generally bark at animate objects), they are similar enough to each other to be reconcilable.

Secondly, given that an expectation failure triggered the learning, and (from the blame assignment phase) given that the failure resulted from the interaction of misindexed knowledge and a novel situation, Meta-AQUA posts a goal to differentiate between the two explanations for why the dog barked (nodes M and M' in Figure 60). Since the conflicting explanations are significantly different (for example, they do not share the same predicate, i.e., detect versus threaten), a knowledge differentiation goal is licensed, rather than a goal to reconcile the two types of explanations. The differentiation goal is achieved if the system can retrieve proper explanations given various situations. The original misunderstanding of the story occurred, not because the explanation that dogs bark when threatened is incorrect in general, but rather because the system did not know the proper conditions under which this explanation applies.

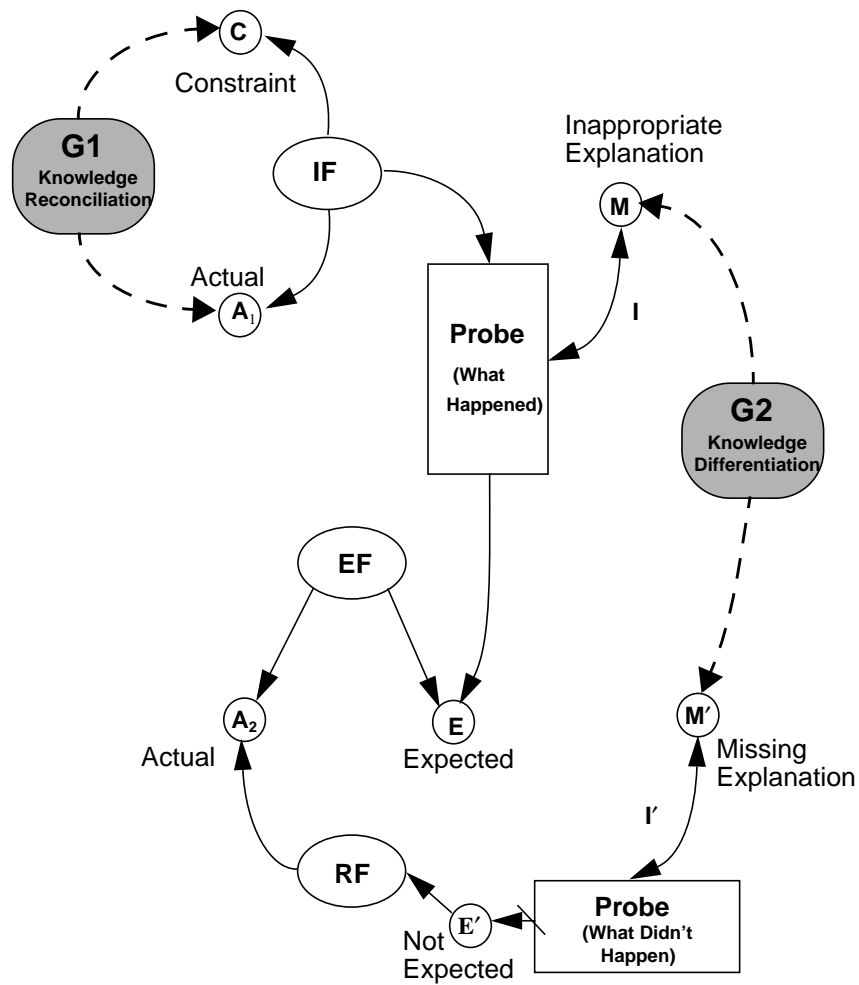


Figure 60. Two learning goals spawned in story HC1

(cf. Figure 55)

In addition to posting these two learning goals, Meta-AQUA places a tentative ordering on their execution (see Figure 56. “Learning output and frame representation of the Meta-XP used in example story” on page 145 for program behavior when deciding what to learn). With no other specific knowledge concerning their respective relations, a good default heuristic is to order them by the temporal sequence of the failures involved in the original reasoning trace. The reason this may be useful is that if it is determined that the first failure was indeed not an error but a misunderstanding or was caused by faulty input, then the reasoning that followed from the first failure (or other assumptions depending on the nature of the first failure that led to the second) may have contributed to the cause of the second. Thus, learning acquired about the first failure may show that the subsequent reasoning was irrelevant, or it may yield more information to be used on the second goal. Therefore, the stage that decides what to learn outputs the knowledge reconciliation goal with priority over the knowledge differentiation goal.

Although learning goals are explicit in the Meta-AQUA system, one should not assume that they are always deliberate goals in cognitive terms; rather, they are as much a computational convenience as a conscious pursuit. As Barsalou (1995) notes, an implicit goal-orientation exists in all learning agents. Thus, one must be careful to distinguish between the computational benefit of expressing goals explicitly and the cognitive interpretation in which some goals may be considered to be either implicit in the behavior (the agent behaves as if having the goal) or subconsciously pursued. I make no claim as to which stance is preferred. For example, it cannot be reasonably claimed that humans actively form a goal to compare visual images, although they constantly do make such comparisons. However, humans can form high-level goals when learning. For instance, novices learning LISP exhibit the goal of trying to understand a programming error by choosing the strategy of re-reading textual instructions or reviewing an earlier example (Pirolli & Recker, 1994; see also Ng & Bereiter, 1991). From a strictly computational perspective, however, the research presented here intends to show that the metaphor of goal-driven planning is a computationally useful one when cast in a learning context. The pursuit of this goal is continued in the next chapter.

## 6.4 Summary and Discussion

This chapter has examined the first half of the hybrid process model of IML theory, concentrating on introspective explanation and the subsequent spawning of learning goals. It has outlined a taxonomy of goal types, including both reasoning and learning goals, and has illustrated these concepts with an example from the Meta-AQUA system.

Although Meta-AQUA is firmly in the CBR tradition, our approach diverges from it somewhat. At least three elements traditionally characterize CBR. First, CBR usually processes instances or concrete episodic cases. However, some systems emphasize the inte-

gration of generalized knowledge and cases (e.g., Aamodt, 1994; Kolodner & Simpson, 1989; Simpson, 1985), and moreover, like Meta-AQUA, some CBR systems actually process abstract cases, including XPs (e.g., Kerner, 1995; Schank et al., 1994). Second, CBR emphasizes the role of memory retrieval of past examples, rather than reasoning from first principles. This focus has led to research on indexing vocabulary and case adaptation. However, Meta-AQUA is a hybrid system that combines the CBR of the first two learning phases with the nonlinear planning of the third. Finally, traditional CBR systems stress goal-directed activity to focus both processing and learning (Kolodner, 1993; Ram & Hunter, 1992; Schank, 1982). Our approach to learning is also goal-directed, but in a very different style. Meta-AQUA is the first CBR system to specifically plan in the knowledge space using goals that specify changes in that space. Unlike INVESTIGATOR (Hunter, 1990a), which creates plans in the external world to achieve learning goals (e.g., access a database to answer a question), Meta-AQUA's plans operate on the internal world of the system's background knowledge.

The next chapter will continue the example to see how Meta-AQUA's learning plans are constructed using a metaphor of nonlinear planning given the learning goals spawned by the case-based method. This is the second part of the hybrid learning model.

## CHAPTER VII

### LEARNING AS A NON-LINEAR PLANNING TASK

*I think people tend to forget that trees are living creatures. They're sort of like dogs. Huge, quiet, motionless dogs, with bark instead of fur.*

—Deep Thoughts  
by Jack Handey.<sup>80</sup>

Humans are amazing in their seemingly effortless ability to learn, yet the objects of learning are quite constrained. For instance, people do not seriously entertain the above analogy that trees are like dogs with bark instead of fur. Furthermore, when most observers read that a dog barks at luggage, they do not infer that the luggage must have been animate, even if the reader has never seen a dog bark at inanimate objects.<sup>81</sup> Instead, if they find the passage odd, they will either implicitly or explicitly question their knowledge of the objects at which dogs bark. One reason for these predispositions is that humans are limited to particular learning spaces.

Children learn very specific conceptual categories of objects in the natural world, such as the distinctions between animate objects, inanimate objects, and functional objects or devices (Keil, 1981, 1989), and they are constrained in the acquisition of word meanings (Carey, 1986). When reasoning and learning, not all possible inferences are warranted or entertained, unlike the approach used in a resolution theorem-prover (e.g., AQ3, Green, 1969). Computational complexity theory has shown that the poverty of stimuli is so pervasive that, for systems to learn at all, they must restrict the kinds of learning that are possible (Osherson, Stob, & Weinstein, 1989; Wexler & Cullicover 1980, both cited in Pylyshyn, 1991). Moreover, Mitchell (1980/1990) argues that to induce anything other than a con-

---

80. Handey (1992).

81. Human readers will entertain such possibilities, however, given the right circumstances. For example, creative understanding requires that the reader suspend judgement when reading science fiction stories (Moorman & Ram, 1994a, 1994b).

junction of all examples in a learner's experience, a learner must possess a learning bias. That is, complex learning is biased toward certain results and only certain learning goals are possible.

The previous chapter presented a taxonomy of learning goals used to provide focus and to keep the learning process tractable. As cognitive science research addresses more sophisticated task domains and as more learning algorithms become available, the associated learning problems become increasingly complex. This focus therefore provides a valuable function. Yet the task of integrating multiple learning methods remains a daunting one because it is an open question as how best to combine the often conflicting learning-methods. This chapter investigates the metaphor of goal-driven planning as a tool for performing such integration. Learning is essentially viewed as solving a planning problem. The planning problem is formulated by posting learning goals (such as the goal to reconcile two divergent assertions with which Meta-AQUA was left at the end of the last chapter) that, if achieved, accomplish some desired change in the system's BK. In response, the learner assembles a learning plan or strategy by choosing and ordering learning methods from its repertoire.

This chapter examines the kinds of interactions that may occur during such construction tasks and how they are managed in the Meta-AQUA system. The example started in the previous chapter (originally from Section 2.1.1) is continued in order to illustrate these interactions. Section 7.1 discusses the issue of whether blame assignment is better directly coupled with learning or mediated with learning through the arbitration of learning goals. Section 7.2 shows how learning strategies can be constructed using nonlinear planning techniques. This section also continues with the Meta-AQUA example examined in the previous chapter. Section 7.3 briefly discusses how learning plans are executed and comments on the usefulness of nonlinear plans. Section 7.4 takes a closer look at the planning metaphor used by the theory of Introspective Multistrategy Learning. Finally, Section 7.5 summarizes and then concludes by briefly discussing related systems, limitations with Meta-AQUA, and areas for future research.

## 7.1 Blame Assignment and Repair: Tight versus loose coupling

The approach of Owens (1990a, 1990b, 1991, 1993) to blame assignment and learning (repair) is very similar to the one presented here. To perform these two tasks in the domain of planning, Owens uses a knowledge structure called a *Plan Failure eXplanation Pattern (PFXP)*. PFXPs are declarative representations for planning failures, each of which is related to a parable such as *too many cooks spoil the broth* but represented in abstract form. They organize high level information about an anticipated problem, lower level details concerning what causes the problem, how to fix it if it happens, and how to learn from it when it occurs. The structures therefore encapsulate and tightly couple the explanations of failure



with those methods used for both plan recovery and plan repair.

Like a Meta-XP, a PFXP is a causal pattern that assists in blame assignment given some symptom of failure. A Meta-XP is a causal structure that connects the symptom to causal factors from the taxonomy of failure causes from Chapter III; whereas, a PFXP is a causal structure that connects a symptom to bad planning decisions such as inappropriate plan transformations. Like Meta-AQUA, Owens' planning system, ANON, uses a characterization of failure to retrieve a PFXP that is used to do blame assignment. Thus, both systems use a case-based method of performing blame assignment. However, unlike Meta-XPs that are organized by theoretical and structural considerations stemming from the underlying models of cognition,<sup>82</sup> PFXPs are organized by the functions they are designed to perform (i.e., by the kinds of recovery and repair operations that will be performed with them). A functional theory (Birnbaum, 1986; Schank, Collins & Hunter, 1986) of blame assignment and learning specifies that a taxonomy should be partitioned depending on what a reasoner does with a member of a category (Owens, 1991). The philosophy of tight coupling (Hammond, 1989; Owens, 1990a, 1990b) dictates that the explanation of failure be closely bound to the repairs and recovery strategies used with the failure. Thus, knowledge used in blame assignment, repair and recovery is cemented into one representation. Tight coupling has a problem, however: repair strategies can interact negatively during learning.

### 7.1.1 Direct Mappings: Tight coupling

Once blame assignment determines an explanation of the fault, given the symptom, a learner must be able to construct a strategy with which to repair the fault. Because a number of faults may occur (corresponding to one or more symptoms), many learning methods or algorithms may be required to repair the knowledge of the reasoner. Just as in blame-assignment, direct linear mappings could be preassigned so that each fault is tightly coupled with one or more repair methods. A large table could implement such a solution by organizing the many-to-many mappings from algorithms that apply in many situations to situations in which a number of algorithms may be relevant (see Table 8). The resulting learning strategy would be an ordered sequence of algorithms selected from the table.

Using this associative matrix technique, a learning system can not only construct a strategy by simply choosing items from the table, but can also optimize the resources expended on the learning. For example, given faults from situations numbered 2 and 3 below, algorithms numbered 1 and 3 can be selected to solve the respective learning prob-

---

82. Table 5, "Detailed taxonomy of causes of reasoning failure," on page 53 comes directly from assumption number one in Figure 40 on page 106. The failure types enumerated in Table 4 on page 50 are derived from the reasoning model of Figure 17 on page 49.

lems. Time and effort can be saved, however, if algorithm number 2 is selected instead, because it provides a repair for both situations.

Table 8: Matrix associative-solution to strategy construction

	Situation 1	Situation 2	Situation 3	Situation 4	.....	Situation M
Algorithm 1		X		X		
Algorithm 2		X	X			
Algorithm 3			X			X
:						
Algorithm N	X					

This type of a solution has been used in a number of theories. For example, CHEF uses configurations of goal failures to directly map to strategies for plan repair (Hammond, 1989), whereas, the TWEAKER module of SWALE uses a taxonomy of XP failure types to directly map to strategies that repair XPs (Kass, 1986). Both systems use direct mapping (indexing) techniques to associate particular failure characterizations to specific repair methods. For instance, the SWALE program explains the unexpected death of a healthy thoroughbred using the “Jim Fixx” XP. Jim Fixx was a world-class athlete who died of a congestive heart failure while jogging. When applying the XP to the death of Swale, the XP fails because it expects Swale to be a human jogger. The failure type NORMATIVE-FILLER-VIOLATION points to a repair strategy that adapts the old explanation to fit the current situation and stores the new explanation for similar future situations (Kass, 1986).<sup>83</sup>

The use of PFXPs improve on this tactic by using the direct mapping approach from an explanation of failure to a repair strategy, rather than from the symptoms of failure to the repair strategy. Moreover, Owens’ use of explanation patterns for blame assignment is an improvement over discrimination nets (e.g., Redmond, 1992) when mapping from symptom to fault (explanation of symptom). But although the principle of tight coupling is useful during blame assignment, it presents problems during learning-strategy selection or construction when multiple faults occur.

---

83. This failure is roughly equivalent to the contradiction failure Meta-AQUA experiences when encountering a dog barking at an inanimate object. The cause is incorrect domain knowledge. Meta-AQUA’s response is to spawn a knowledge reconciliation goal (as seen in the last chapter). This chapter will address the issue of what occurs in response to the learning goal.

### 7.1.2 Indirect Mappings: Loose coupling

The issue involved with using a direct-mapping solution for the learning-strategy construction problem is that repair methods may actually interact in such a manner that one method negatively affects the outcome of another. It is unreasonable to assume that all learning algorithms are functionally independent, especially considering that they were designed and implemented in isolation without regard to integration issues. Therefore, when a learner chooses two or more repair methods, the order of the methods may have an undesirable effect on the results of learning. Analogous problems have certainly been shown to exist with respect to intelligent planners; this section demonstrates that the same type of problems occur with multistrategy learning, and so a loose coupling is preferred.

Even if a system's goals to learn are implicit in the behavior of the system rather than explicit, because numerous faults may co-occur, the repairs chosen to correct the faults are bound to interact in ways that are difficult to predict. In formulations with conjunctive goals, numerous difficulties arise such as goal interactions, protection intervals, and many-to-many relationships between goals and algorithms. For instance, a familiar goal conflict in planning systems is the "brother-clobbers-brother" goal interaction (Sussman, 1975) whereby the result of one plan that achieves a particular goal undoes the result or precondition of another plan serving a different goal. If a learning goal specifies a state change to the background knowledge of the system, rather than a state change in the world, then learning plans can have similar effects. Changes to specific knowledge may affect previous changes to the background knowledge performed by other learning algorithms. For example, any change to a conceptual attribute must occur before a separate algorithm uses that same attribute in index learning, otherwise the index might become obsolete.<sup>84</sup>

Therefore, the tight coupling approach used with PFXPs has significant disadvantages. A PFXP must contain not only pointers to the repair strategies that are relevant to particular failure patterns, but it must also contain information concerning how these strategies are interleaved. The issue must also be addressed concerning what happens when two or more PFXPs both apply (e.g., how is the learning managed when both *too many cooks spoil the broth* and *you can catch more flies with honey than you can with vinegar* apply to a given situation). If the interactions involved are captured in more complex PFXP types and if the researcher does not anticipate all of them, then the burden of acquiring these composite patterns is upon the learning system. Such interactions are managed naturally with a nonlinear planner when only the constituent failure types and the associated learning goals are specified by the researcher.<sup>85</sup> Even if the tight coupling approach can acquire information about interactions, such a system must dynamically monitor its learning for

---

84. Section 7.2.4 will argue this point more vigorously in the context of a concrete example.

goal violations (a process that is explicitly performed by standard planners).

The next section presents evidence from a hand-coded example to show that nonlinear planning techniques can eliminate the negative effects present from interacting learning methods. This lends support for the position that to include a separate learning stage that posts explicit learning goals (i.e., that decides deliberately what to learn) is an improvement over learning systems that do not use such an intermediate stage. Chapter IX presents a more careful empirical study which reaches the same conclusion.

## 7.2 Learning-Strategy Construction: Learning as a planning task

The strategy-construction stage of learning builds a learning plan to achieve the learning goals posted by the stage that decides what to learn. The construction entails not only choosing and ordering the algorithms and operators to achieve the learning goals, but if needed, it implies spawning subgoals for learning goals that cannot be achieved directly.

An important issue in the strategy-construction problem is the manner in which learning plans are created (Hunter, 1990b; Redmond, 1992). There are two approaches to this issue: a system can have either a static or a dynamic planner. A static planner simply uses the characterization of the learning goal and the context (explanations produced by the blame assignment phase) as an index into a memory of stereotypical plans. Once retrieved, a learning plan is instantiated and parameterized by the context, and then executed. More flexible and complex, the dynamic approach performs goal-subgoaling to produce plans rather than simply using canned or hand-tailored plans. The approach here is to use the dynamic method.

This section examines the strategy-construction problem by first discussing how a nonlinear planner handles goal interactions (Section 7.2.1) and then by looking at how this is accomplished in a small domain such as the blocks world (Section 7.2.2). Subsequently, the section returns to the drug-bust example to see how Meta-AQUA performs the same tasks, but in the space of changes to the BK using learning goals. The last chapter showed how Meta-AQUA spawned two such learning goals. This chapter shows how plans can be constructed for the knowledge differentiation goal (Section 7.2.3) and the knowledge reconciliation goal (Section 7.2.4). The next major section (Section 7.3) briefly discusses how these plans are carried out.

---

85. Section 7.4.2, “Advantages of the Planning Metaphor,” starting on page 171 enumerates a number of additional properties that make the use of learning goals suitable to the task of constructing a learning strategy or selecting a repair.

Q1: How to construct a learning strategy?

Ans1: Treat strategy construction as a planning problem.

### 7.2.1 Managing Goal Interactions with Nonlinear Planning

Hierarchical non-linear planners like NOAH (Sacerdoti, 1975) and Nonlin (Ghosh et al., 1992; Tate, 1976) are designed for the task of noticing the interactions between cognitive processes. A hierarchical planner creates an abstract plan to achieve a conjunction of goals and then successively refines the steps until the plan terminates in primitive actions that can be directly executed. The planning technique is nonlinear because it uses a least-commitment approach that assumes all steps can co-occur unless otherwise required. For this reason, steps in the plan may follow a partial order, rather than a full linear order. Two steps will be placed in a particular order with respect to each other, if and only if they conflict or interact in some (positive or negative) manner.

The Task Formalism language of Tate (1976) specifies that schemas are of two basic types. Operator schemas (opschemas) define planning operators and action schemas (actschemas) define actions in the representation hierarchy of a given domain. An opschema captures the way in which a goal is achieved, whereas an actschema represents the effect of an action in the representation. Each schema is represented with the following basic attributes (fields).

- The *todo* field is a pattern to be matched when expanding an abstract step into a refined form. When a pattern is matched it is replaced with the steps in the *expansion* field.
- The *expansion* field lists a partially ordered sequence of substeps to accomplish the goal in the *todo* field.
- The *orderings* field specifies any ordering enforced on the steps in the *expansion* field.
- The *conditions* field determines when a matched schema is applicable. Four types of conditions exist (*precond*, *unsuperv*, *use-when*, and *use-only-for-query*) as explained below.

- The *effects* field determines the effects of the operator or action in the manner of STRIPS (Fikes & Nilsson, 1971). It contains an addlist and a delete list which creates new states and removes old ones from the environment respectively.

The conditions field of an action or operator definition is used to establish the states under which the schema should be used. The *precond* condition type is equivalent to Tate's supervised condition and represents the standard notion of a plan precondition. They specify the previous states that need to exist before the planner applies the schema. If the states do not exist, then the planner creates a subgoal to achieve them. The *use-when* condition is also called a filter condition, and was termed a hold condition by Tate. Converse & Hammond (1992) call them appropriateness conditions and directs special attention to the distinction between them and normal preconditions above. They represent the conditions that should exist for the schema to apply to the current needs of the planner, rather than simply those conditions that must exist for the operator to be used. Unsupervised conditions are specified as type *unsuperv*. They are used to place a linear order on plan steps, rather than expand steps. Finally, the *use-only-for-query* type of condition is used to bind variable temporarily.

### 7.2.2 Planning for Blocks World Goals

As an example of interacting goals, consider the state of Figure 61 when a planner is given the goals of arranging block A on block B and block B on block C. To place A on B, block A must first be clear and then it is moved on top of block B. To achieve the goal of B on C, B only need be moved. However, if the order chosen is to move the block B first, then the act of clearing A in preparation of its move to B will undo the goal state of having B on C. Therefore, the order of achieving the two initial goals are important because the results of the operations used to carry out the plans do interact.

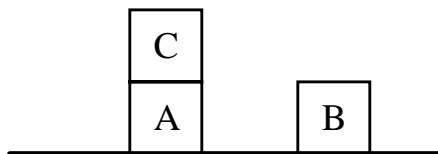


Figure 61. The Sussman anomaly

Figure 62 contains the Task Formalism definitions for the operators and actions that stack blocks and that manage interactions such as the one discussed above. Given the goals (on A B) and (on B C), a planner will match the *todo* pattern of operator *makeon*. Note

```

(opschema makeon
:todo (on ?x ?y)
:expansion (
  (step1 :goal (cleartop ?x))
  (step2 :goal (cleartop ?y))
  (step3 :action (puton ?x ?y))
)
:orderings ( (step1 -> step3) ( step2 -> step3))
:variables (?x ?y)
)

(opschema makeclear
:todo (cleartop ?x)
:expansion (
  (step1 :goal (cleartop ?y))
  (step2 :action (puton ?y ?z))
)
:orderings ((step1 -> step2))
:conditions (
  (:use-when (on ?y ?x) :at step2)
  (:use-when (cleartop ?z) :at step2)
  (:use-when (not (equal ?z ?y)) :at step1)
  (:use-when (not (equal ?x ?z)) :at step1)
)
:variables (?x ?y ?z)
)

(actschema puton
:todo (puton ?x ?y)
:expansion ( (step1 :primitive (puton-action ?x ?y)))
:conditions (
  (:use-when (cleartop ?x) :at step1)
  (:use-when (cleartop ?y) :at step1)
  (:use-only-for-query (on ?x ?z) :at step1)
)
:effects (
  (step1 :assert (on ?x ?y))
  (step1 :assert (cleartop ?z))
  (step1 :delete (cleartop ?y))
  (step1 :delete (on ?x ?z))
)
:variables (?x ?y ?z)
)

```

---

Figure 62. Blocks world operators

that to achieve the goal, the planner will replace the goal with the steps in the `expansion` field of the operator. These steps are partially ordered so that both steps one and two must come before step three, but the operator assumes that both the first two can be executed in parallel. It is only when the planner discovers the interaction between steps to achieve these goals, that one order will be chosen over another.

Using these same principles, the research here has defined operators for learning methods that solve learning goals in the BK, rather than object configuration goals in the blocks world. Instead of primitive steps that are assumed to be implemented by a robot arm (e.g., the `puton-action` step of schema `puton`), the primitive steps of learning operators represent calls to standard learning algorithms. The following section makes this analogy more clear by returning to the drug-bust example started in the previous chapter.

### 7.2.3 Planning for a Knowledge Differentiation Goal

Blame assignment during the drug-bust example retrieved an IMXP that explained the faulty explanation for why the dog barked at the luggage in the airport. Instead of barking because it was threatened by the luggage, the dog barked because it detected contraband in the luggage. The system then spawned two learning goals: G1, a knowledge reconciliation goal, and G2, a knowledge differentiation goal (Figure 63). The learner thus must reconcile the input (previously believed to be faulty) with its conceptual definition of dog-barking, and it must differentiate the two explanations so that neither is confused for the other.

Consider the knowledge differentiation goal, G2, of Figure 63. It seeks to differentiate between the expected explanation that the dog barked because it was threatened and the actual explanation that the dog barked because it detected contraband. This goal can be achieved by reindexing the memory locations for the two explanations so that they will be retrieved when appropriate. However, because the system has no prior experience with the actual explanation,  $A_2$ , (and thus the system neither foresaw nor considered the explanation), the learner posts a subgoal to expand the instantiated explanation (i.e., the knowledge expansion goal G3) to produce the missing explanation pattern,  $M'$ .

The schemas that produce the subgoal sequencing of events are defined in Figure 64. The `use-when` conditions on the `index-xp` operator guarantees that this schema is chosen only if the variable `?x` is both an XP and is a token (instance) rather than an type (abstract explanation). The `gen-op` operator does not actually decide which algorithm to perform the generalization; the `do-generalize` action schema does. Explanation-based generalization (EBG) (DeJong & Mooney, 1986; Mitchell, Keller & Kedar-Cabelli, 1986) can be selected as an appropriate learning algorithm for this task.



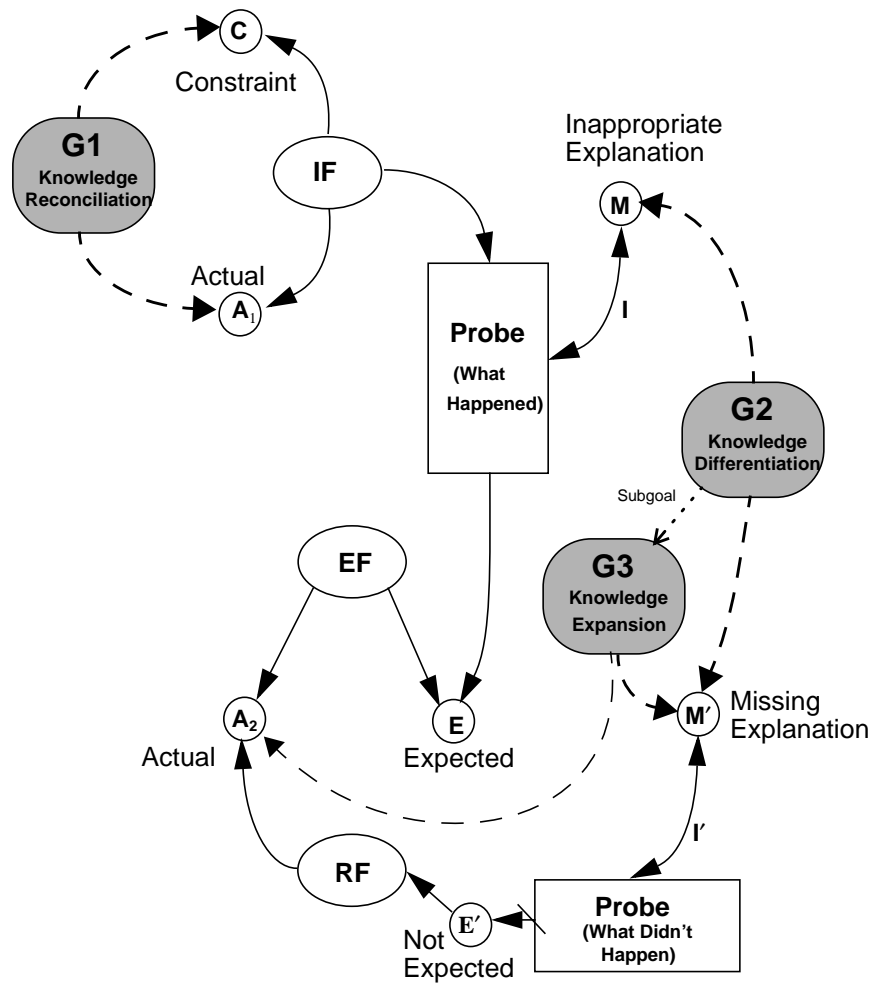


Figure 63. Abstracted IMXP with learning goals

```

;;; To index in memory some XP that is a token,
;;; first spawn a subgoal to expand the concept
;;; into a type, then perform the (non-primitive)
;;; action to index the XP.
;;;
(opschema index-xp-op
  :todo (indexed ?x)
  :expansion (
    (step1 :goal (knowledge-expansion ?x))
    (step2 :action (index-item ?x))
  )
  :conditions (
    (:use-when (isa xp ?x) :at step1))
    (:use-when (isa token ?x) :at step1)
  :orderings ( (step1 -> step2) )
  :variables (?x)
)

;;; To achieve a knowledge expansion goal,
;;; perform some kind of generalization.
;;;
(opschema gen-op
  :todo (knowledge-expansion ?x)
  :expansion (
    (step1 :action (do-generalize ?x))
  )
  :variables (?x)
)

```

---

Figure 64. Schema definitions to index an XP

A more difficult problem is to differentiate the applicability conditions for the two abstract explanations ( $M'$ , the one produced by generalizing the detection explanation,  $A_2$ , and  $M$ , the original XP that produced the initial threaten explanation,  $E$ ) by modifying the indexes ( $I'$  and  $I$ ) with which the system retrieves those explanations. If the two problems of erroneous association and novel situation were to be treated independently, rather than as a problem of interaction, then an indexing algorithm would not be able to ensure that the two explanations would remain distinct in the future. That is, if the learner simply detects a novel situation and automatically generalizes it, then indexes it by the salient or causal features in the explanation, and if the learner independently detects an erroneous retrieval, and re-indexes it so that the same context will not retrieve it in the future, then there is no guarantee that the resultant indexes will be mutually exclusive. Instead, the system must re-index  $M$  *with respect to*  $M'$ , not simply with respect to the condition with which  $M$  was retrieved. Therefore, the problems associated with direct mapping from blame assignment to strategy construction without the mediation of learning goals are substantial.

The problems to be solved, then, are determining the difference between  $M$  and  $M'$ , and, in the light of such differences, computing the minimal specialization of the index of  $M$  and the maximally general index of  $M'$  so they will be retrieved separately in the future. In the case of the drug-bust story, HC1, the problem is somewhat simplified. The difference is that retrieval based on the actor relation of barking actions (dogs) is too general. The threaten explanation applies when dogs bark at animate objects, while the detection expla-

nation is appropriate when dogs bark at containers.

Figure 65, “Mutual-indexing schemas,” shows learning-operator definitions for the indexing strategy that manages mutual indexing between two concepts. The operator schema determines that both items must be independently indexed before they are indexed with respect to each other. The action schema has filter conditions (`use-when`) that apply when both are indexed and both are XPs. An unsupervised condition (`unsuperv`) specifies that if there exists a change in the explained action, then it must occur before the execution of this schema. That is, a linearization must be performed on external goals to reorder any other schema that may establish the change. It says in effect that we want all attributes of the target concept to be stable before it operates on the concept; no other operator can change an attribute in order for the changes performed by indexing to be unaffected.

#### 7.2.4 Planning for a Knowledge Reconciliation Goal

In Figure 63 on page 163, the remaining learning goal, G1, represents a knowledge reconciliation goal. The goal is to reconcile the fact that the conceptual definition of dog-barking is limited to animate objects with the fact that a particular dog barked at a piece of luggage. This goal can be thought of as a simple request for similarity-based learning (SBL) or inductive learning (e.g., UNIMEM’s SBL algorithm in Lebowitz, 1987, or abstraction transmutation as in Michalski, 1994). The system is simply adding an additional positive example to the instances seen. An incremental algorithm is required because this instance has been discovered after an initial concept has been established some time in the past.

An interesting interaction can occur, however, if the system waits for the result of the EBG algorithm required by the knowledge-expansion subgoal spawned by the knowledge-differentiation goal discussed above. The algorithm will generalize the explanation (that this particular dog barked at a particular piece of luggage because it detected marijuana) to a broader explanation (that dogs in general may bark at any container when they detect contraband). Thus, the example provided to the inductive algorithm can be more widely interpreted, perhaps allowing its inductive bias to generalize the constraint, C, on the object of dog-barking to `physical-object` (the exhaustive case of `animate-object` and `inanimate-object`), whereas a single instance of a particular breed of dog barking at a specific brand of luggage, A<sub>1</sub>, may limit the inductive inference if no additional domain knowledge is available.

Unfortunately, however, because the EBG algorithm uses the representation of the dog-bark definition, and the inductive algorithm changes this definition, the induction must occur first. Thus, the system cannot take advantage of the opportunity cited in the previous

```

;;; To index two items with respect to each other,
;;; make sure that they are both indexed independently,
;;; then index them jointly.
;;;
(schema mutual-index-op
  :todo (index-wrt-item ?x ?y)
  :expansion (
    (step1 :goal (indexed ?x))
    (step2 :goal (indexed ?y))
    (step3 :action
      (index-dual-items ?x ?y)))
  :orderings(
    (step1 -> step3)
    (step2 -> step3))
  :conditions (
    (:precond (indexed ?x)
      :at step3 :from step1)
    (:precond (indexed ?y)
      :at step3 :from step2)
    (:use-when (not (equal ?x ?y))
      :at step1))
  :effects ( )
  :variables (?x ?y))

;;; The action of indexing two explanations jointly requires
;;; that any changes to the definition of the parent type
;;; of the domains of their explains-node (that is, the
;;; explained-action) be performed before the indexing is.
;;;
(schema do-mutual-xp-indexing
  :todo (index-dual-items ?x ?y)
  :expansion ( (step1 :primitive
    (perform-mutual-indexing ?x ?y)))
  :conditions (
    (:use-when (indexed ?x) :at step1)
    (:use-when (indexed ?y) :at step1)
    (:use-when (isa xp ?x) :at step1)
    (:use-only-for-query
      (explains ?explains-node ?x)
      :at step1)
    (:use-only-for-query
      (domain ?explains-node
        ?explained-action)
      :at step1)
    (:unsuperv (changed true ?explained-action)
      :at step1)
    )
  :effects (
    (step1 :assert (indexed-wrt ?x ?y))
    (step1 :assert (indexed-wrt ?y ?x)))
  :variables (?x ?y ?explains-node ?explained-action))

```

---

Figure 65. Mutual-indexing schemas

paragraph. One important implication of this point is that in systems which plan to learn, if the reasoner does not anticipate this second interaction (thus placing EBG before the induction), the system must be able to perform dynamic backtracking on its decisions.

Like a non-linear planner in the blocks world, the learning system must detect any dependency relationships so that goal violations can be avoided. For example, when the definition of dog-barking is modified by generalizing the constraint on the objects at which dogs bark from `animate-object` to `physical-object`, any indexing based on the modified attribute must occur after this modification, rather than before it, to avoid indexing with obsolete conceptual knowledge.<sup>86</sup>

Note that the action schema of abstraction in Figure 66 has an `addlist` and `delete list` effect that modifies the `changed` predicate on the domain of the constraint relation. This effect, along with the unsupervised condition of the `do-mutual-xp-indexing` action schema of Figure 65, determines that the indexing will not be performed until the constraint becomes stable. Therefore, if both schemas are being instantiated, the `Nonlin` module of `Meta-AQUA` will automatically order the abstraction before the indexing. A similar unsupervised condition prevents generalization of the detection explanation from occurring before the concept of dog-barking is stable.

```
;; Perform an abstraction transmutation on relation r1 given relation r2.
;; The function raises the co-domain of r1 to the shared parent type of
;; r1 and r2.
;;
(actschema do-abstraction-change
:todo (abstracted ?r1 ?r2)
:expansion ( (step1 :primitive (perform-abstraction ?r1 ?r2)))
:conditions (
  (:use-when (isa relation ?r1) :at step1)
  (:use-when (isa relation ?r2) :at step1)
  (:use-when (relation ?r1 ?r1-type) :at step1)
  (:use-when (relation ?r2 ?r2-type) :at step1)
  (:use-only-for-query (domain ?r1 ?r1-domain) :at step1)
  (:use-only-for-query (co-domain ?r1 ?c) :at step1)
  (:use-only-for-query (co-domain ?r2 ?a) :at step1)
  (:use-only-for-query (parent-of ?c ?c-parent) :at step1)
  (:use-only-for-query (parent-of ?a ?a-parent) :at step1)
  (:use-when (equal ?r1-type ?r2-type) :at step1)
  (:use-when (equal ?c-parent ?a-parent) :at step1))
:effects (
  (step1 :assert (co-domain ?r1 ?c-parent))
  (step1 :assert (changed true ?r1-domain))
  (step1 :delete (co-domain ?r1 ?c))
  (step1 :delete (changed false ?r1-domain)))
:variables (?r1 ?r2 ?r1-type ?r2-type ?r1-domain ?c ?a ?c-parent ?a-parent))
```

---

Figure 66. Abstraction schema

---

86. This result supersedes the conjecture by Ram & Hunter (1992) that, unlike standard planning techniques, interactions and dependencies do not occur with learning goals.

### 7.3 Strategy Execution: Performing the learning and the aftermath

After a learning plan is constructed, a very simple process can execute the plan. All primitive steps in the plan are calls to learning algorithms from the toolbox. Because the plans are partially ordered, not all steps will have a linear order enforced. Therefore, some steps may be executed in parallel. In the drug-bust example, however, the final learning plan Meta-AQUA constructs is fully ordered. As shown by the Nonlin output in Figure 67, the resultant steps are (1) perform an abstraction transmutation on the concept of dog barking (realizing that dogs bark at containers); (2) perform EBG on the new explanation (producing a generalized version); (3) index the generalized XP in isolation; and finally, (4) use the new concept definition to mutually differentiate and index the two generalized explanations of why dogs bark. This plan is translated back into a frame representation and executed in the order specified.

After the learning is performed, control is returned to the story understanding module. The system continues with the story until completion. In subsequent stories (or as will be seen in the next chapter, even within the same story), the same types of failures should not repeat if the learning is successful. For example, after Meta-AQUA finishes reading story HC1, it can understand the story in Figure 68 correctly. In this story a police officer and a canine enter a suspect's house, the dog barks at a garbage pail, and the suspect is arrested for possession of some marijuana found in the pail. The new story causes no anomaly when the dog barks at the inanimate container. Indeed, Meta-AQUA expects some type of contraband to be found in the container after it reads that the dog barked, but before it is told of the contraband's existence in the story. Thus, learning accomplished in the previous story improves both understanding of and predictions for subsequent stories.

S1: A policeman entered a house of a suspect  
 S2: A police dog entered the house with him.  
 S3: Dog barked at the suspect's garbage pail.  
 S4: The dog barked because it detected a half a  
 kilogram of marijuana in the pail.

---

Figure 68. Story HC3: Another hidden stash

### 7.4 The Planning Metaphor in Learning

The idea of applying the metaphor of goal-directed planning to learning tasks presents a number of interesting research issues. The planning community has investigated prob-

```

UMCP NONLIN V1.2 (11/91).
****
The world state is ((CO-DOMAIN BARK-EXAMPLE INANIMATE-OBJ) (DOMAIN BARK-EXAMPLE DOG-BARK) (RELATION BARK-EXAMPLE OBJECT) (ISA RELATION BARK-EXAMPLE) (CO-DOMAIN BARK-DEF ANIMATE-OBJ) (DOMAIN BARK-DEF DOG-BARK) (RELATION BARK-DEF OBJECT) (ISA RELATION BARK-DEF) (DOMAIN ACTOR2 DOG-BARK) (EXPLAINS ACTOR2 DETECT-XP) (RELATION THREATEN-XP CAUSAL-RELATION) (ISA XP THREATEN-XP) (INDEXED THREATEN-XP) (ISA TYPE THREATEN-XP) (ISA RELATION THREATEN-XP) (DOMAIN ACTOR1 DOG-BARK) (EXPLAINS ACTOR1 DETECT-XP) (RELATION DETECT-XP BECAUSE) (ISA XP DETECT-XP) (ISA TOKEN DETECT-XP) (ISA RELATION DETECT-XP) (ISA CASE CASE1) (ISA TOKEN CASE1) (INDEXED CASE2) (ISA CASE CASE2) (ISA TYPE CASE2)).
The problem to be solved is:
{SCH10158}
  TEST::PLAN-SCHEMA
    Expansion:
      0 {<ND10150>[:DUMMY]}
      1 {<ND10151>[:GOAL(KNOWLEDGE-RECONCILIATION-GOAL MIKE BARK-DEF BARK-EXAMPLE)]}
      2 {<ND10152>[:GOAL(KNOWLEDGE-DIFFERENTIATION-GOAL MIKE DETECT-XP THREATEN-XP)]}
      3 {<ND10153>[:DUMMY]}
    Conditions:
      <<SC10154>> <<SC10155>>
    Effects:
      <<SE10156>> <<SE10157>>
Attempting to establish any unsupervised conditions...
Done establishing any unsupervised conditions.
***** PLANNING COMPLETED *****

*** The (current) plan is:
===== INITIAL STATE =====
(CO-DOMAIN BARK-EXAMPLE INANIMATE-OBJ)
(DOMAIN BARK-EXAMPLE DOG-BARK)
(RELATION BARK-EXAMPLE OBJECT)
(ISA RELATION BARK-EXAMPLE)
(RELATION THREATEN-XP CAUSAL-RELATION)
(ISA XP THREATEN-XP)
(INDEXED THREATEN-XP)
(ISA TYPE THREATEN-XP)
(ISA RELATION THREATEN-XP)
(DOMAIN ACTOR1 DOG-BARK)
(EXPLAINS ACTOR1 DETECT-XP)
(RELATION DETECT-XP BECAUSE)
(ISA XP DETECT-XP)
(ISA RELATION DETECT-XP)
(ISA CASE CASE1)
(ISA TOKEN CASE1)
(INDEXED CASE2)
(ISA CASE CASE2)
(ISA TYPE CASE2)
(ISA TOKEN DETECT-XP)
(CO-DOMAIN BARK-DEF ANIMATE-OBJ)
(PARENT-OF INANIMATE-OBJ PHYSICAL-OBJ)
(PARENT-OF ANIMATE-OBJ PHYSICAL-OBJ)
(ISA PERSON MIKE)

===== PLAN ACTIONS =====

3: :PRIMITIVE (ABSTRACT-ITEM BARK-DEF BARK-EXAMPLE)

8: :PRIMITIVE (EBG DETECT-XP)

6: :PRIMITIVE (PERFORM-INDEXING DOG-BARK)

4: :PRIMITIVE (PERFORM-MUTUAL-INDEXING DETECT-XP THREATEN-XP)

=====

```

---

Figure 67. Nonlin output and the final learning-plan

lems of goal interaction, uncertainty, strategy selection, error recovery (including backtracking during the planning process and rollback during plan execution), and concurrency. In one form or another, all of these issues reappear given a learning interpretation. Broadly construed, the technique of nonlinear planning in the pursuit of explicit goals can be directly mapped to learning. Instead of desired states in the world, learning goals represent desired states in the background knowledge of the learner. Instead of operators that result in actions performed by agents, learning operators result in actions by learning algorithms.<sup>87</sup> However, at a finer level of granularity the metaphor may not map so neatly. For example, this research has established that the brother-clobbers-brother goal interaction is present in some situations during multistrategy learning; yet, it is not immediately apparent whether or not all types of goal interactions from the classical planning literature will apply to operators executing in the background knowledge. Therefore, one of our future research goals is to more fully determine where the planning metaphor fits a learning framework and under what conditions it does not.

#### 7.4.1 Generality of the Metaphor

Moreover, the learning performed by Meta-AQUA is not tied to either the task of story understanding or the domain of criminal activities. By adding a few conceptual definitions for a new domain, Meta-AQUA processes and learns from the following story that parallels story HC1 of Section 7.2.

S1: A person enters the handball court.  
 S2: The person suddenly hit a handball.  
 S3: He hit the ball because he wanted to have fun.

---

Figure 69. Story HC1': The handball game

As before, S1 is skimmed and, because Meta-AQUA believes that people hit animate objects, S2 generates an anomaly. It explains the anomaly by concluding that the person is trying to hurt the ball. When given a new explanation, Meta-AQUA generalizes it, indexes the new explanation with respect to the hurt explanation, and loosens the constraint on the object of *hit* to include toys as well as animate objects. Meta-AQUA uses the same Meta-XP as a pattern of failure and guide to learning as in the previous story.

---

87. See also Etzioni, Hanks, Weld, Draper, Lesh, & Williamson (1992) for related work on formalizing the notion of information goals that can be pursued by planners.



### 7.4.2 Advantages of the Planning Metaphor

The planning and problem-solving literature suggest that use of explicit goals have many benefits over *ad hoc* processing. Many of these benefits apply to learning-goal processing as well as standard-goal processing. Some of the advantages of using learning goals to mediate between the blame-assignment and strategy-construction stages are as follows:

- *Decouples the many-to-many relationship between failure and repair*

For a given failure there may be more than one algorithm which needs to be applied for learning. Conversely, a given algorithm may apply to many different types of failures. A direct mapping from blame or fault to algorithm choice is more difficult and less flexible than the use of learning goals.

- *Allows an opportunistic approach to solving learning problems*

It is not always guaranteed that sufficient resources and/or knowledge are available to perform learning at the time that a system realizes that it needs to learn. When this condition occurs it is possible to index the learning goal in memory so that it can be retrieved at a later time when these requirements are met.

- *Allows chaining, composition, and optimization of the means by which learning goals are achieved*

For example, one algorithm may achieve two or more goals, whereas in other cases, many strategies may apply to a single goal. If more than one plan applies, a system should use the one which may contribute to the maximum achievement of other goals with the minimum amount of resource consumption.

- *Allows detection of dependency relationships so that goal violations can be avoided*

As described by this chapter, it is important to recognize that when multiple items are learned from a single episode, the changes resulting from one learning algorithm may affect the knowledge structures used by another algorithm. Such dependencies destroy any implicit assumption of independence built into a given learning algorithm used in isolation. For example, the definition of dogs barking is modified by Meta-AQUA so that its constraint on those objects at which dogs bark is generalized to `physical-object` from `animate-object`. However, any indexing based on the attribute associated with the objects of barking dogs must occur after this modification, rather than before it, to avoid indexing on obsolete conceptual knowledge.

- *Allows parallel execution of learning algorithms*

Because nonlinear plans specify a (minimal) partial ordering of steps, learning algorithms may be executed concurrently if no ordering is imposed between two or more steps by goal interactions or subgoal refinement. The issue of concurrent execution of learning algorithms is virtually unaddressed in the machine learning community, but potentially of great computational benefit. We raise the issue here, but the full exploration of such opportunities awaits further research.

## 7.5 Summary and Discussion

This chapter has covered a number of diverse issues relating to the problem of learning-strategy construction. Converging evidence argues that the learning task can successfully be treated as a nonlinear planning task. The first section of the chapter debated that a loose coupling between blame assignment and learning is preferred over tight coupling. The next section explained how a nonlinear planner can perform this function, given the example from the last chapter. Subsequently, a section explained how learning plans are executed. Finally the advantages of using the metaphor of planning was enumerated.

In review, a number of important knowledge dependencies were illustrated using the drug-bust example. These are enumerated below.

- Even after learning goals are spawned when deciding what to learn, dynamic monitoring of goals sometimes requires subgoals to be spawned. This occurred when the knowledge expansion goal was made a subgoal to the knowledge reconciliation goal.
- If direct mapping were to be used, reindexing would not make the associations of two confused explanations different with respect to each other. They would be indexed independently.
- Knowledge dependencies can occur when planning to learn in the BK. The indexing of a concept depends on the implicit assumption that the attributes by which indexing is performed remain stable.
- An additional dependency exists such that changes to a conceptual definition must occur before generalizing the concept. That is, it was necessary to change the constraint on object attribute of bark before generalizing bark itself.

Introspective reasoning is crucial to detecting these dependencies. Although many

computational systems use a reflective reasoning approach (e.g., Collins, Birnbaum, Krulwich, & Freed, 1993; Fox & Leake, 1994; Oehlmann, Edwards, & Sleeman, 1994; Plaza & Arcos, 1993; Stroulia & Goel, 1995), and a few have used the planning metaphor in learning (Hunter, 1990b; Quilici, in press; Ram & Hunter, 1992; Ram & Leake, 1995; Redmond, 1992), none of these systems have applied the planning metaphor as strictly as Meta-AQUA has; none execute a planner like Nonlin upon its own knowledge. One important implication of this approach is that nonlinear learning plans can take advantage of the inherent parallelism associated with learning problems. As machine learning algorithms increasingly enter the applied world, the need for parallelism will become evermore important. Future research will better illuminate the potential of this insight.

Additional future research must be directed toward incorporating more learning strategies. One of the weak points of the current system is that it reasons during learning at a macro-level. Meta-AQUA recognizes the functional difference between generalization and specialization and therefore can choose an appropriate algorithm based on which algorithm is most appropriate. However, it cannot currently select between competing algorithms that both perform generalization. Meta-AQUA does not reason at the micro-level, as do systems that address the selective-superiority problem<sup>88</sup> in inductive learning (see, for instance, Brodley, 1993; Provost & Buchanan, 1992; Schaffer, 1993), although the scope of learning problems solved by Meta-AQUA is greater than these other systems.

Another limitation of the Meta-AQUA implementation is that learning self-evaluation (step 3 of Figure 48, “IML learning algorithm,” on page 126) does not exist. Thus, Meta-AQUA cannot cross-validate or compare various successful algorithms, nor can it currently judge when learning fails and another algorithm must be chosen. Just as it detects, explains, repairs and learns from reasoning failures, an interesting line of future research would be to allow Meta-AQUA to reason about its own learning. See Leake (1992) for approaches to this problem.

To perform multistrategy learning, an intelligent system must consider a number of factors that are not significant in isolated learning systems. In particular, a system must be able to handle insufficient resources and knowledge and manage dependency relations between learning algorithms at run-time. Many alternative solutions and interactions may occur, even when reasoning about simple situations. Treating the learner as a planner is a principled way of confronting these difficulties. Many of the techniques and results from the planning literature<sup>89</sup> can be appropriated in learning systems to provide a better level of

---

88. Empirical results suggest that various inductive algorithms are better at classifying specific classes or particular distributions of data than others. Each algorithm is good at some but not all learning tasks. The selective superiority problem is to choose the most appropriate inductive algorithm, given a particular set of data (Brodley, 1993).

robustness and coverage in situations where many types of failure may occur. The aim is to transform these failures into opportunities to learn and improve the system's overall performance.

---

89. One interesting approach to dealing with task interaction in planning domains appears in Freed & Collins (1994). To repair its own planning mechanisms, Freed and Collins show that a learner can use self-knowledge along with an understanding of its planning failures that result from task interactions in the performance domain.

***Part Four***

***IMPLEMENTATION AND CONCLUSION***



## CHAPTER VIII

### META-AQUA

*Total grandeur of a total edifice,  
Chosen by an inquisitor of structures  
For himself. He stops upon this threshold  
As if the design of all his words takes form  
And frame from thinking and is realized.*

—Wallace Stevens (1952), pp. 510-511.

The Meta-AQUA system implements the theory of introspective multistrategy learning presented in the previous chapters by providing a computational realization of the concepts within the theory. The project of building this implementation has been especially challenging. Not only does the Meta-AQUA system, like many other programs, have a performance system that manipulates an explicit representation of the world, but in addition it has a learning system that analyzes and learns from the results of the performance. These results, along with a trace of the performance itself, must also be represented explicitly. Moreover, the system uses three different representational formalisms: frames, CDs, and predicate logic. The implementation is hence naturally complex.

As a result of the system's complexity, deliberate decisions were made to implement only the most significant portions of the theory within the Meta-AQUA program. For example, the memory system in Meta-AQUA is not a full model of human memory. Instead, it is a rough approximation to the indexed dynamic memory as described in Schank (1982). Although the implementation of the memory is crude compared to the elaborate functionality of similar memories in other programs (e.g., CYRUS - Kolodner, 1984; DMAP - Martin, 1990; ANON - Owens, 1990a; SMART - Veloso & Carbonell, 1990), it still is a better cognitive model of memory than those using exhaustive search (e.g., FUNES - Markovitch & Scott, 1988).

Also, as a matter of pragmatics, the Meta-AQUA implementation does not assume that all possible causes for failures enumerated in Chapter IV can occur. Instead the pro-

gram restricts the number of causes it considers during blame assignment.<sup>90</sup> Although future research intends to expand the scope of blame assignment, the current implementation still far exceeds the number of causes that related computational systems entertain. For example, the MINERVA learning system (Park & Wilkins, 1990) is a theory revision system similar to Meta-AQUA in its use of explanation and introspection. However, because MINERVA assumes a consistent knowledge base, brute force search, no input, perfect processes, and well-behaved goals, missing domain knowledge is the only possible cause of failure. By making such assumptions, the blame-assignment task is circumvented altogether. All failures are attributed to missing pieces of domain knowledge in its background knowledge.

A few minor differences exist between IML theory and the embodiment of the theory in the Meta-AQUA system. These discrepancies will be made explicit in this chapter. Notwithstanding these differences, the implementation represents a substantial undertaking and will be examined in some detail. The initial section (8.1) outlines Meta-AQUA's system architecture and its file system. Subsequent sections examine the performance subsystem (8.2), the input problem generator (8.3), the memory system (8.4), and the learning subsystem (8.5), each in turn. Examples of the running behavior of the system when processing automatically generated input stories (rather than hand-coded stories) illustrate many sections throughout. The final section (8.6) closes the chapter with a summary and a brief discussion.

## 8.1 Meta-AQUA System Architecture

Meta-AQUA is a learning system that chooses and combines multiple learning methods from a toolbox of algorithms in order to repair faulty components responsible for failures encountered during the system's performance task. The program incorporates an introspective version of the AQUA (Ram, 1991, 1993, 1994) story-understanding system as the performance task from which learning can take place. As a front end module to the performance system, a specially modified version of the Tale-Spin (Meehan, 1981) story-generation program automatically produces input data. At the back end, the UM Nonlin planning system (Ghosh et al., 1992) creates a learning plan designed to improve the performance. An extensive frame system (Minsky, 1975; Wilensky, 1986b) was built to provide the formalism with which to represent the system's knowledge, both of the domain and of itself. This knowledge is stored in and retrieved from a simple indexed memory. The memory is partitioned into a working memory (FK) and a long-term store (BK).

---

90. See Section 8.5 for a specification of the scope of these assumptions with respect to the failure causes of Table 5, "Detailed taxonomy of causes of reasoning failure," on page 53.



The system architecture and flow of information within Meta-AQUA is shown in Figure 70. The problem generation module outputs a story to the performance system with the initial goal to understand the input (i.e., build a coherent conceptual interpretation). The story understanding system uses schemas from the BK to build a representation of the story in the FK. If this task fails, then a trace of the reasoning that preceded the failure is passed to the learning subsystem. A CBR subsystem within the learner uses past cases of introspective reasoning from the BK to explain the failure and to generate a set of learning goals. These goals, along with the trace, are then passed to a nonlinear planner. The planner subsequently builds a learning strategy from its toolbox of learning methods. The learning plan is then passed to an execution system that examines and changes items in the BK. These changes enable improved performance in subsequent processing.

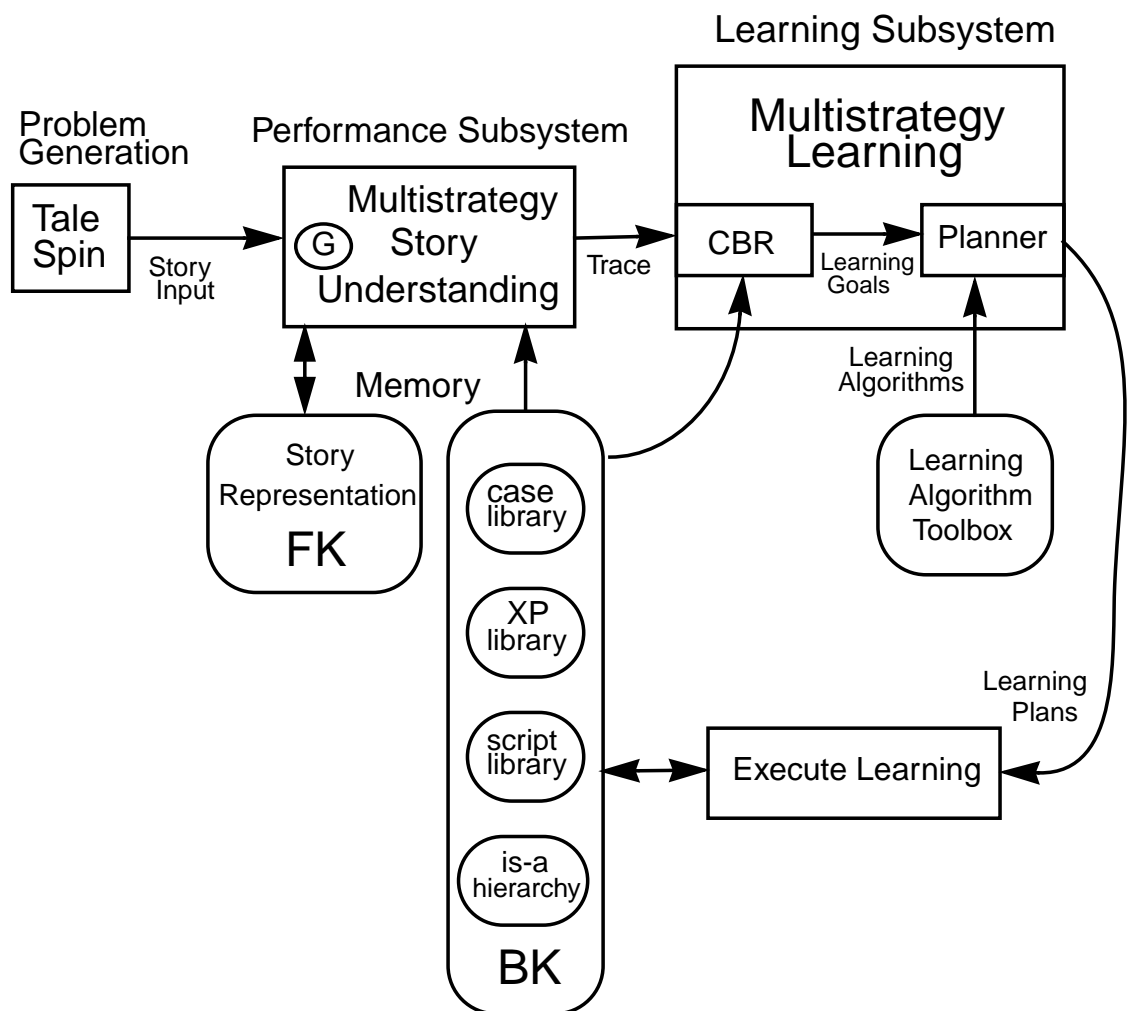


Figure 70. Detailed Meta-AQUA system architecture

Meta-AQUA is programmed in Symbolics Common LISP under the Genera operating system (Version 8.3). The hardware platform is a Symbolics MacIvory Model-3 LISP microprocessor embedded in a Macintosh IIfx personal computer. Including comments and documentation, the LISP source code takes up approximately 750 kilobytes of disk space in sixty-seven files. The file system definition is pictured in Figure 71.<sup>91</sup> Two of the modules (`frame` and `non-lin`) are stand-alone subsystems each of which have their own `defsystem` constructs. The four other modules are native to the Meta-AQUA system. The five main modules (not including documentation) each have a separate symbol package. The following subsections describe each of the major subsystems in turn.

## 8.2 The Performance Subsystem

The AQUA program is a question-driven story understanding system whose task is to explain terrorist activities and events contained in newspaper-like stories provided as input. The Meta-AQUA system learns about drug-smuggling activities, given AQUA's prior experience with stories about terrorists. Both systems' performance task is to "understand" stories by building causal explanations that link the individual events into a coherent whole.<sup>92</sup> Meta-AQUA adds introspective reasoning and multistrategy learning using Meta-XP structures and the learning theory presented in parts Two and Three of this thesis. In addition and unlike AQUA, the performance sub-system of Meta-AQUA uses a multistrategy approach to understanding. Thus, the top-level goal is to choose a comprehension method by which it can understand the input.

To process input information, the system posts all performance goals and learning goals in a priority queue. These goals are then processed by the current priority value such that the highest value is pursued first. For instance, when a new conceptual input arrives, a goal to understand the input is placed in the priority queue with a nominal value. If no other goals have higher values, then the comprehension goal will be taken from the queue and processed. The goal to understand an input is then made into a subgoal to detect an

---

91. This definition does not include some minor details such as the statements that determine module dependencies (i.e., `:in-order-to` clauses).

92. Meta-AQUA actually has three performance modes from which it can operate. This chapter discusses Meta-AQUA performance in `read-story` mode (story understanding). But in addition, the system can be set to a `LISP-programming` mode in which the system attempts to create and understand LISP code (see Section 9.3). A third mode, `act-out-story`, simulates goal-driven planning behavior between two interacting characters. The characters represent a custom official and a smuggler who attempt to act out the dynamics of story HC1 in the airport terminal. Although this partially implemented mode is mostly outside of the scope of this document, see Chapter XIII for speculations regarding the interaction of planning and understanding and the potential contribution of this performance mode within the system.

```

(defsystem Meta-AQUA

  ;; The documentation for the Meta-AQUA System
  (:module docs ("Meta-AQUA.doc") (:type :text))

  ;; The FrameSystem is a stand-alone subsystem for representation. (nine files)
  (:module frames Frame-System (:type :system))

  ;; The knowledge representation definitions. Written in frame notation.
  (:module representations
    ("reps-import.lisp"           ;; Module interface specification
     "rep_smuggle4.lisp"         ;; AQUA knowledge from terrorist domain
     "rep_meta-xps.lisp"         ;; XP and Meta-XP representations
     "rep_planner.lisp"          ;; Knowledge to support act-out-story mode
     "rep_lisp-programmer2.lisp"  ;; Knowledge to support LISP mode
     "rep_tspin.lisp"            ;; Knowledge to support Elvis World (Tale-Spin)
     "rep_hit.lisp"              ;; Knowledge to support sports events
     "rep_scripts.lisp"          ;; Script representations
     "rep_cop_scripts.lisp"       ;; Particular scripts to understand police activities
     "rep_roles.lisp"))          ;; Miscellaneous relations

  ;; Nonlin is a stand-alone subsystem. (nineteen files)
  (:module non-lin Nonlin-System (:type :system))

  ;; Meta-AQUA proper.
  (:module meta-aqua
    ("meta-aqua-interface.lisp"   ;; Module interface specification
     "constants.lisp"            ;; Global constants
     "lowlevel.lisp"             ;; Low-level functions for screen handling, etc.
     "story-input.lisp"          ;; Hand-crafted input examples
     "goal-q.lisp"               ;; Goal priority-queue management
     "memory.lisp"               ;; Memory management
     "meta-xp-procs.lisp"        ;; Procedures for handling meta-xps
     "learner.lisp"              ;; Learning library and associated functions
     "script-applier.lisp"       ;; Simple script applier
     "questions.lisp"            ;; Question handling
     "explainer.lisp"            ;; Explanation facility
     "cbr.lisp"                  ;; Simple case-based reasoner
     "understander.lisp"         ;; Main story-understanding code
     "solver.lisp"               ;; Main code for problem solving mode
     "eval.lisp"                 ;; Functions to calculate learning performance
     "init.lisp"                 ;; Initialization functions
     "main.lisp"))              ;; Main control loop and synonym functions

  ;; The Tale-Spin automatic story generator.
  (:module tale-spin
    ("spin-interface.lisp"       ;; Module interface specification.
     "tspin.lisp"                ;; The main control functions.
     "extensions.lisp"           ;; New functionality for Tale-Spin.
     "spin-cd-reps.lisp"         ;; CD representations for Elvis World.
     "data.lisp"                 ;; Initial states of the world.
     "patch.lisp"                ;; A fix to the main function.
     "mumble.lisp"               ;; CD to English translator.
     "verbs.lisp"                ;; Verb definitions.
     "alt-story.lisp"            ;; Alternate initial states.
     "tspin-tokens-4-meta-aqua.lisp" ;; Frame representations for all tokens.
     "tspin-2-meta-aqua.lisp"))  ;; CD to frame conversion functions.
)

```

---

Figure 71. Meta-AQUA file system definition

anomaly in the input.

For the task of story understanding, Meta-AQUA employs an algorithm whose flow of control is outlined in Figure 72. First, the outer loop inputs a sentence representation and checks to see if the concept can answer a prior question. If it can, the reasoning associated with the question is resumed. Otherwise, the concept is passed to the understanding algorithm. The understanding algorithm consists of three phases: anomaly identification, hypothesis generation, and hypothesis verification.

The first phase looks for questions associated with the concept by checking the concept for interesting characteristics. Meta-AQUA considers acts of sex, violence, and loud noises inherently interesting (Schank & Abelson, 1977). Moreover, any concept that is anomalous is considered interesting (Ram, 1990b), as is any concept about which the program has recently learned something. Inherently interesting acts are detected by the concept type of the input. Anomaly checking is performed by comparing the input to the conceptual definitions found in the conceptual hierarchy. If a concept contradicts a constraint, an anomaly exists and a question is posed. Such a question represents a fundamental learning goal (or more specifically, a knowledge acquisition goal). The goal is to construct or otherwise acquire an explanation of why the anomaly exists. If no anomaly is detected, the concept is skimmed. Control then passes back to the beginning.

When an input is skimmed, it is passed to a simplified version of SAM, a script application program (Cullingford, 1978, 1981). The script applier understands a story by matching input sentences to stereotypical sequences of events (i.e., to scripts). For example, the simple **drug-bust** script consists of an initial drug detection, then confiscation of the contraband, followed by the arrest of the person possessing the drugs. Although scripts omit many of the causal relations between events in a story, they can help an understander interpret a story by providing details not explicitly mentioned in the story. During the skimming process, however, if an input is not matched with a script, then the input is placed on a list of current story structures in the FK to await further processing.

As an example of the role scripts play in the input to the understanding process, consider the following. The **pipe-smoking-script** contains an instrumental scene (**gain-control-of-object**) that establishes the preconditions (possession of a pipe) necessary for the goal scene (**smoke-pipe**) of the script. The instrumental scene itself has sub-scenes that must be matched to the input, some of which in turn may have additional sub-scenes. They include the sub-scene **open-container** (if the pipe is in the cupboard), an **ATRANS** (to transfer possession to the smoker), and **close-container**. But in the event of a match, not only are the sub-events which came from the story examined by the understander to detect anomalies, but the inferred **gain-control-of-object** structure is input for anomaly detection as well.

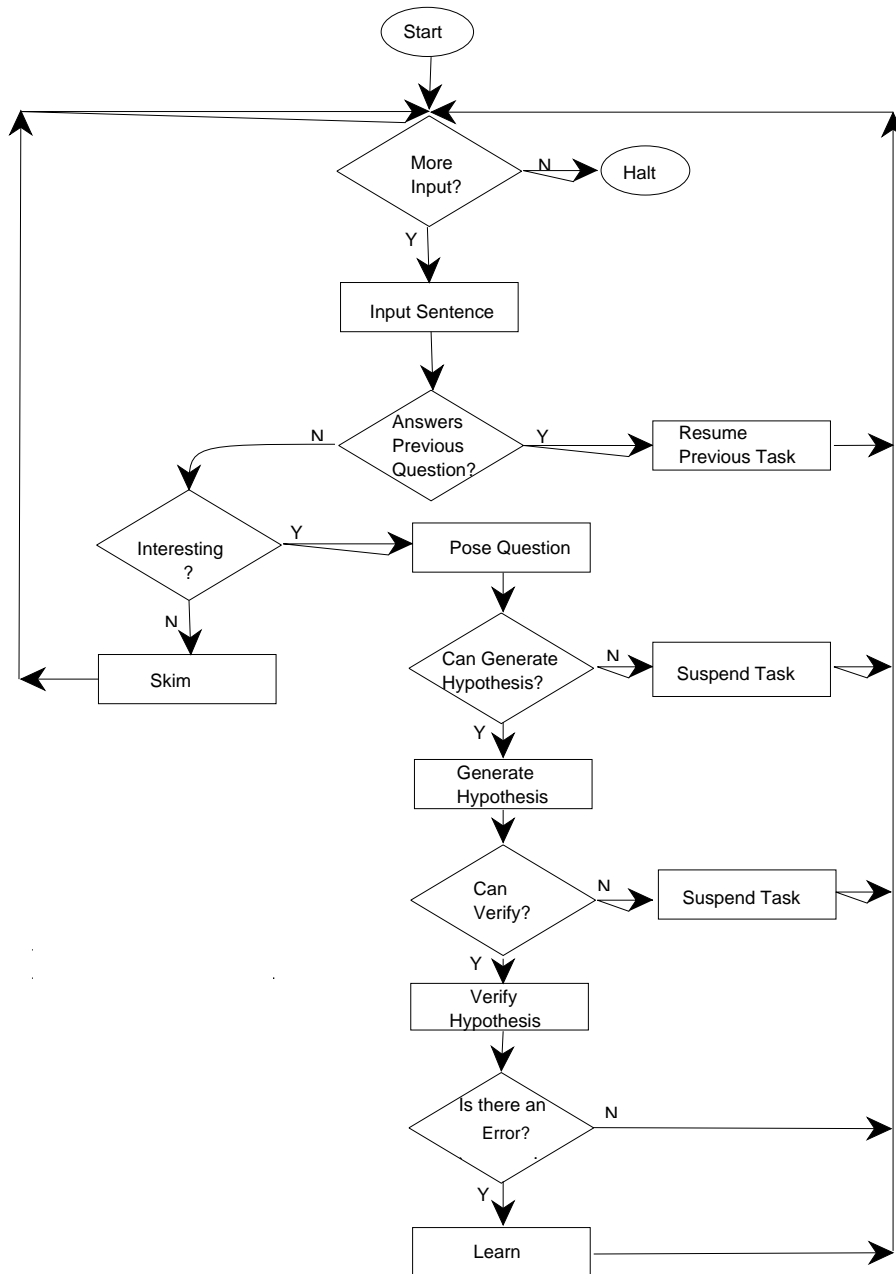


Figure 72. Meta-AQUA flow of control

Rather than skimming the input, the system may examine it in more detail by posing questions about particular parts of the input. If a question is posed, the understander attempts to answer the question by generating a hypothesis. The basis of the decision to pose a question (i.e., what knowledge is relevant in making the determination) is then recorded in a TMXP. Strategies for hypothesis generation include application of known explanation patterns (“XP application”), case-based reasoning, and analogy. If none of these methods applies, then the process is suspended until a later opportunity presents itself.<sup>93</sup>

After a hypothesis is generated, the potential answer passes to the verification phase of the performance task. Strategies for hypothesis verification include devising a test (currently not implemented), comparison to known concepts, and suspension of the reasoning task. Following this, the system reviews the chain of reasoning during the verification phase to detect failure. The failure detection process examines the reasoning trace using the algorithm described by Figure 47 on page 121. If a failure occurs, then control passes to the learning subsystem and further input processing is suspended until control returns.

## 8.3 Input Subsystem

Input to the performance system can originate in two different ways. Hand-coded stories provide explicit demonstrations of specific features of the IML learning algorithm. A number of these stories (e.g., HC1 and HC2) have been discussed in previous chapters. In addition, a modified version of the Tale-Spin (Meehan, 1981) story-generator was integrated into the Meta-AQUA architecture to produce stories automatically. The function of a separate generator is to attenuate the bias of the programmer who creates hand-coded stories and to provide a means for randomly varying the conditions under which learning takes place. These two conditions provided by the generator have enabled the design of an empirical study that will be reported in the next chapter.

### 8.3.1 The Tale-Spin Story-Generator

Given a main character and a problem for the character, the Tale-Spin module creates stories by simulating the actions that would be necessary for the character to achieve goals stemming from the problem. For example, if a character has the problem of being thirsty, Tale-Spin assigns the character an initial goal to remove a state of thirst. The character can achieve the goal by travelling to where water or drink exists if the location is known. If it is not known, the character can ask another agent in the story. The character then

---

93. Section 8.4 briefly describes how such processes are resumed.

gains possession of the drink and finally ingests it. At selected points, the generator inserts random events into the story. For each event in the story, the generator adds any associated causal results from the event. These results change the world and enable further actions by the characters in the story. For example, the act of travelling to the location of water enables the taking possession of it which in turn enables the drinking of it. This final action removes the hunger. The story terminates when the goals and subgoals of the main character have been achieved or when all possible plans to achieve them have been exhausted.

The Tale-Spin program was obtained from the University of California at Irvine<sup>94</sup> where it is used as a problem generator for the OCCAM learning system (Pazzani, 1994). OCCAM learns about physical causation given stories in which characters perform actions such as playing ball when bored. So for example, when children accidentally drop a solid ball, it will not break; whereas, when children drop a balloon upon a sharp object (e.g., a rose bush), the object will break. In this case, OCCAM learns the causal interaction between object composition and surface impact. In OCCAM's world, four main characters exist. They are Dad, Mom, and their two children Lynn and Lynn. The problems that they encounter are hunger, thirst and boredom. In addition, the house contains a cat who randomly knocks vases from tables to the floor. Given these initial program conditions, Tale-Spin was extended to produce stories in the domain of criminal activities.

### 8.3.2 The Elvis World

In order to support large data collection, additions to Tale-Spin provide numerous scenarios with a potentially infinite number of variations that test Meta-AQUA's ability to learn from explanation failure. Among the changes, a musician named Elvis and a police officer were added to the cast of characters. Elvis is temporarily boarding with Mom, Dad and family, whereas the officer occasionally visits the house, presumably because of neighborhood complaints of loud music and raucous behavior. Furthermore, the police officer often (but not always) brings a drug-detection dog along with him, and the domestic household now contains a pet dog.

Two new problem types were also added. Characters may be *jonesing*<sup>95</sup> for drugs. In Elvis' case, he sometimes smokes marijuana to relieve his jones, whereas Dad occasionally smokes a pipe with tobacco (see Figure 15 on page 43). Lynn has also been given

---

94. At URL <ftp://ics.uci.edu/pub/machine-learning-programs/TalespinOccam>, the code is publicly available through the World Wide Web.

95. In the vernacular, a "jones" is a drug habit accompanied by withdrawal symptoms. The verb "to jones" is to be going through a state of withdrawal.

a tobacco habit. The police officer has the problem of being *concerned* about the law. The state of being concerned is relieved if he can either locate contraband or arrest criminals.<sup>96</sup> The program was also modified to hide the marijuana during story initialization in different locations around the house (e.g, in the cupboard, refrigerator, and under the carpet), so the officer's task varies depending on entry conditions (i.e., at what point in the story the officer arrives on the scene and whether the dog accompanies him), the initial location of the pot, and the actions of the characters in the story.

Moreover, to facilitate the performance task, the Tale-Spin program was modified so as to generate explanations of key events in the stories. The resolution of all anomalies are thus incorporated within every story. For example, Tale-Spin always includes a reason why police dogs bark when generating a story. Although in an ideal implementation, the understanding process should be able to make powerful enough inferences to confirm explanations of the input independently, the performance task has been simplified within Meta-AQUA. Instead of using inference to confirm hypotheses, Meta-AQUA mainly depends on the story to provide explanations that confirm them. For the implementation, the research goal is to concentrate on the learning task rather than the understanding task.

An example of Tale-Spin output is shown in the story TS1 of Figure 73. The example is roughly equivalent to the hand-coded story that Chapters VI and VII examined in detail (i.e., the airport drug-bust story, HC1, shown in Figure 52 on page 140). Figure 74 shows the specific sentences of story TS1 that correspond to the sentences of story HC1. Unlike the hand-tailored stories, however, the length of Tale-Spin's stories range from 3 to 108 sentences and average approximately 30.

Unlike AQUA, the Meta-AQUA story understanding subsystem does not actually parse the sentences from an English representation. Because the focus of this research does not center on the natural language understanding problem, Meta-AQUA assumes that input sentences are already represented conceptually (i.e., Tale-Spin pre-parses them). The *mumble* module of Tale-Spin provides stylized English paraphrases to assist the system user.

---

96. Unlike the UC Irvine version of Tale-Spin in which characters and their goals did not interact, the program has been modified so that the police officer is a competing character with his own problem and goal as he arrives on the scene. Because the police will confiscate the marijuana when found and then arrest Elvis, such events may preempt the enabling conditions of actions Elvis had planned to perform. For instance, if Elvis is thirsty but the officer arrests him, this condition restricts his freedom of movement so that he cannot go to the faucet for water. Therefore, the story can end with Elvis still having the problem with which he began (i.e, thirst).



One day Elvis was jonesing. Elvis took the lighter1 from the table2. He had the lighter1. The table2 didn't have the lighter1. Police and dogs arrived. The phone1 was ringing. Mom picked up phone-receiver1. The phone1 wasn't ringing. She had phone-receiver1. She let go of phone-receiver1. She didn't have phone-receiver1. Officer1 went to outside. She [Mom] pushed light-switch1. The light1 was on. The cat1 pushed the vase2 to the floor1. The vase2 was broken. The police-dog1 went to outside. He [Officer1] pushed door-bell-switch1. The door-bell1 was ringing. He didn't push door-bell-switch1. The door-bell1 wasn't ringing. He went to the kitchen. The police-dog1 went to the kitchen. The police-dog1 went to the vase3. (S25) The police-dog1 sniffed the vase3. (S26) The police-dog1 barked at the vase3. The police-dog1 was barking. He [Officer1] went to the vase3. He took the ganja1 from the vase3. He had the ganja1. The vase3 didn't have the ganja1. (S32) He arrested Elvis. He controlled Elvis. He went to outside. Elvis went to outside. The police-dog1 went to outside. (S37) If the police-dog1 detects the ganja1 then the police-dog1 will bark at the vase3. He [Elvis] was still jonesing.

--- The End ---

---

Figure 73. Tale-Spin story TS1  
(Annotations in brackets; sentence numbers in parentheses)

S25: The police-dog1 sniffed the vase3.

S26: The police-dog1 barked at the vase3.

S32: Officer1 arrested Elvis.

S37: If the police-dog1 detects the ganja1 then the police-dog1 will bark at the vase3.

---

Figure 74. Sentences from story TS1 corresponding to HC1

### 8.3.3 Interface to the Performance System

As seen in Figure 75, the interface between Tale-Spin and the performance system is simple. The `spin` function of Tale-Spin takes a character and problem to generate a CD representation of the story. The `mumble` function generates the English equivalent of the CD conceptual representation. Both of these outputs are then placed on the global variable `*ALL*`. This variable is a list of tuples of the form `<CD “generated equivalent text”>`. A translation routine (function `convert-story`) then converts the CDs into a frame representation used by Meta-AQUA. The result is placed on the global list `*Story-Concepts*`. This structure is a list of tuples of the form `<frame “equivalent text”>`. Then for each input concept, the function `init-goals` creates a knowledge acquisition goal to understand the concept. The function places each goal on the `*Goal-Queue*` priority queue. In turn, the performance system evaluates each input, places the interpreted result on the variable `*World-Model*` (so it can easily be displayed at the end of the program as output), and indexes the result in the FK.

Before we examine the memory system, it should be made explicit that although there is a tight interface between the Tale-Spin story generator and the Meta-AQUA story understanding subsystem, the implementation of and theory behind the Tale-Spin generator in no way reflects the content or claims concerning IML theory. On the contrary, the manner in which input is created for the performance system is largely incidental to the theory and the implementation of the rest of the Meta-AQUA system. Meehan (1981) asserts that stories should be both interesting and coherent. Interesting stories set up a focal problem domain that span a number of levels. Coherent stories contain rational characters that pursue individual goals that then interact. Moreover, Meehan claimed that story simulation itself was a form of cognition (p. 203). None of these theoretical underpinnings directly impinge on the theory of learning presented here. We do not subscribe to nor refute any such claims. The program is used simply as a matter of computational convenience and to generate less biased input.

## 8.4 Memory

Computer memory is often viewed as a virtually error-free medium in which retrieval of data is performed by simple fetch operations. As computer memories grow, however, brute-force search for the address to perform the fetch becomes increasingly intractable. Memory indexing is added in order to make memory retrieval more efficient. A memory-indexing mechanism is a trade-off between time to search and accuracy of retrieval; though efficiency is gained, poor indexing schemes risk not finding the proper information. Indexes are pointers from some feature in the environment (cue) to a memory element associated with that feature.

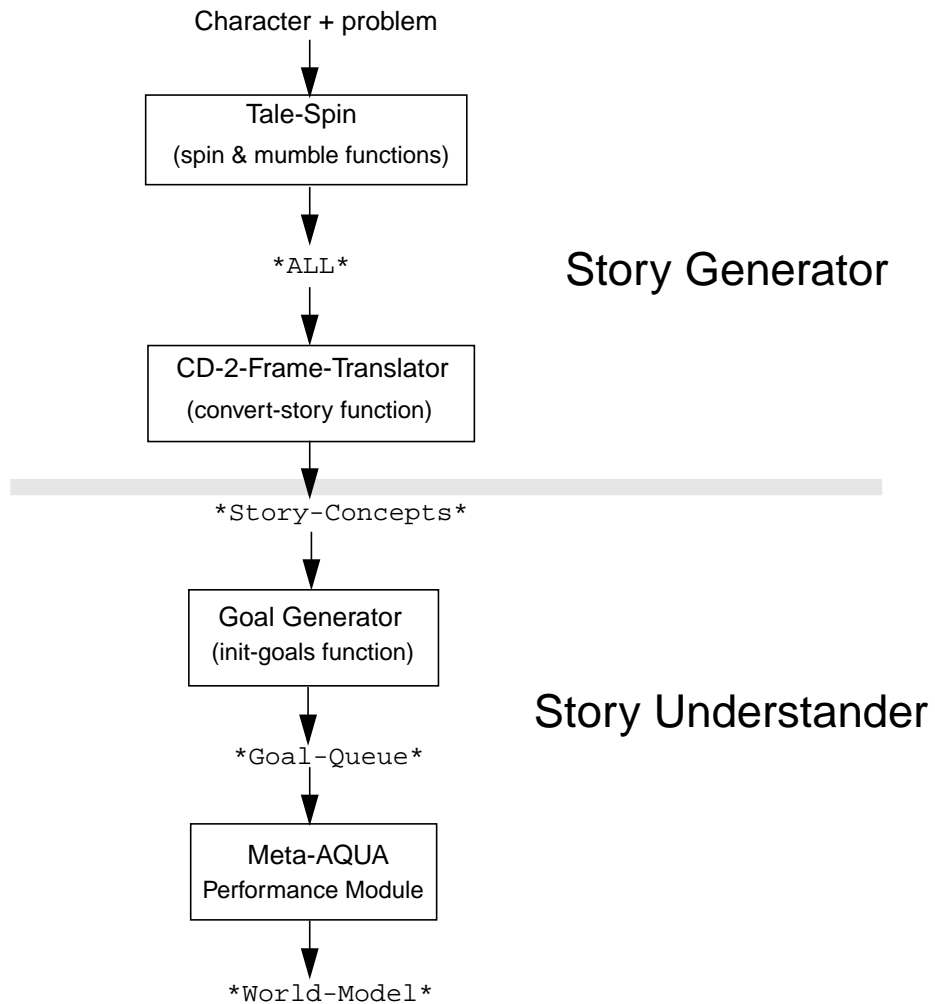


Figure 75. Representational flow between generator and understander

The memory for Meta-AQUA is partitioned into two indexed memories: a working memory called the foreground knowledge, or FK, and a long-term store called the background knowledge, or BK. The BK used in the current implementation consists of a multiple-inheritance conceptual hierarchy, a case library of past episodes, a set of story scripts, and an indexed collection of XPs. The FK is a dynamic list of structures representing the current understanding of the story or “world model.” Although both memories are implemented as lists of LISP symbols that have particular values, all processes other than printing and bookkeeping functions access memory by matching the cues chosen at retrieval time with the indexes created at storage time.

This section explains the representation used for items stored in memory (Section 8.4.1), describes the implementation used for indexing items in memory and in retrieving them (Section 8.4.2), and works through a short story that Tale-Spin generates in order to demonstrate the advantages of opportunistic memory in the Meta-AQUA system (Section 8.4.3). The representation, indexing scheme and the structure of the memory subsystem allows the generation of multiple hypotheses and their respective verification to be interleaved in arbitrary order. The example also shows that Meta-AQUA can benefit from learning even before it is finished processing the story in which the learning takes place.

### 8.4.1 Types and Tokens

The distinction between a *type* and a *token* is an important difference maintained in Meta-AQUA’s memory. Types represent conceptual categories and are defined through the frame-based knowledge-representation system underlying the Meta-AQUA program. Tokens are instantiated instances (reified types). For example, Meta-AQUA’s BK contains a type definition associated with the act of barking by dogs. This type captures in general the assumed constraint that dogs bark at animate objects (see page 56 for an abbreviated type definition for *dog-barks*). Contrastingly, as Meta-AQUA receives input, it instantiates a token for the each event or assertion, including the episode of a particular dog barking at a particular inanimate object. These tokens are incorporated into retrieved cases, scripts, or XPs and stored in the FK.

A token is represented in memory as a *frame variable*. A frame variable is a unique LISP symbol, such as the gensym `XP-GOAL-OF-OUTCOME->ACTOR.115`.<sup>97</sup> The symbol value of a frame variable is either a *frame form* or a *literal frame*. A literal frame is an arbitrary structured symbol-value, whereas a frame form is represented as a list con-

---

97. Such an XP explains that a particular actor chose to perform a particular action because of the goal to achieve a state for which the action results. See Figure 22 on page 76 for a representation of such XPs.

sisting of a *frame-type designator* followed by zero or more *slots* (see Figure 76). Literals are distinguished from other frame variables in that routines which traverse frame structures cannot inspect or otherwise traverse a literal.<sup>98</sup> Slots are attribute-value relations (sometimes called slot-filler pairs) having a particular name (i.e., the *role* of the slot) and an arbitrary number of *facets*. A distinguished facet for every slot is the *value facet*, representing the *role-filler* (or simply *filler*) of the slot. Facet values may be either an *attribute value*, a frame, or list of frames. Attribute values are special terminal frames that specify a member of an enumerated set. For example, **green.0** is an attribute value of the enumerated type color-value. It fills the value facet of a color slot and evaluates to the slotless frame form “(green)”. As mentioned in Section 4.4, slots themselves are treated in the frame system as a first-class objects, and thus have explicit frame representations (see Figure 25., “Relations as first-class objects,” starting on page 81). Thus, the *relation facet* has as its filler a frame representing the attribute-value relation.<sup>99</sup>

```

Frame-Token.772 <-
(FRAME-TYPE
  (attribute-1  (facet-1-1 value-1-1)
                (facet-1-2 value-1-2)
                ...
                (facet-1-i value-1-i))
  (attribute-2  (facet-2-1 value-2-1)
                (facet-2-2 value-2-2)
                ...
                (facet-2-j value-2-j))
  ...
  (attribute-n  (facet-n-1 value-n-1)
                (facet-n-2 value-n-2)
                ...
                (facet-n-k value-n-k)))

```

---

Figure 76. Generalized frame token structure

A special property of the knowledge representation system is that type definitions (categories) are stored as the symbol values of the frame-type designator. As a result, every token has close proximity to its corresponding type. Given a frame token as shown in Figure 76, the type definition can be found by taking the symbol-value of the symbol

---

98. The reason for this property is that literals have no sub-frame value.

99. For details about similar approaches to knowledge representation, see Jones (1992), Ram (1989), and Wilensky (1986b).

'FRAME-TYPE. Thus in general, to find the conceptual definition of an arbitrary frame variable, X, the LISP call (symbol-value (first (symbol-value X))) will suffice.<sup>100</sup>

### 8.4.2 Indexing

Conceptually, an index is a mapping from a context to a memory item. Indexes describe a situation in a story or in reasoning about a story and are typed according to the role they play in reasoning. Depending on the index type (*question-type.0*, *xp-type.0*, *case-type.0*, or *plan-type.0*), indexes map to different conceptual memory items (questions, xps, cases and plans, respectively). In Meta-AQUA, each index is composed of simple chains of *micro-indexes* (see Bhatta, 1995; Kolodner, 1993, pp. 193-245; and Owens, 1993 for descriptions of more sophisticated index implementations).

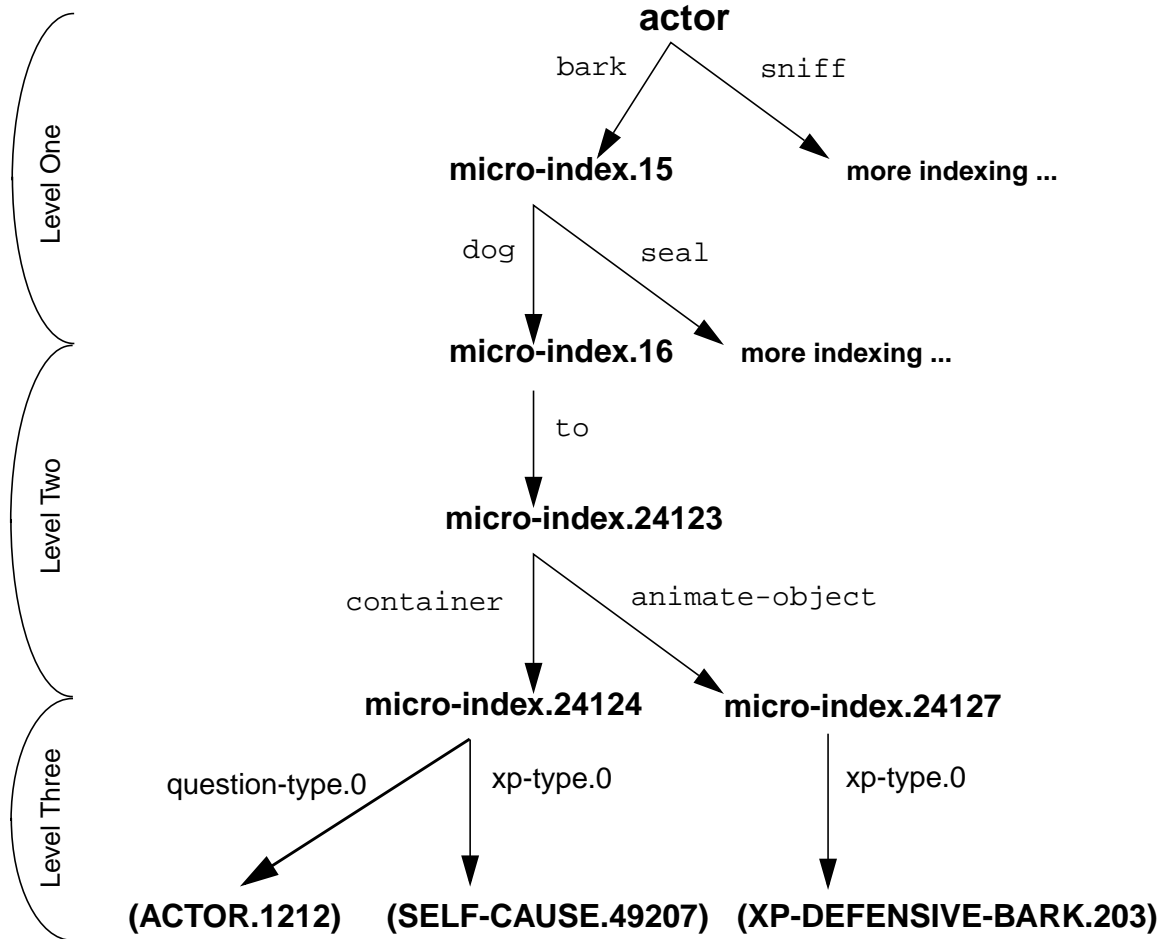
For example when processing story TS1, Meta-AQUA poses the question “why did the dog bark at the vase?” It hypothesizes that the vase somehow threatens the police dog, but because it cannot verify this, the system suspends the deliberation and indexes the question into memory. The index used to store the question is an index of type *question-type.0* based on a dog being the actor of the barking event and vase being the object. The choice of features is determined by the path along which the anomaly is discovered in the input structure. When it later examines the explanation that the dog barked because it detected drugs, the input causes a reminding that resumes the prior verification process. The composition of these types of indexes is shown in Figure 77.

Indexing structure in Meta-AQUA is composed of three levels. The primary level of all indexing is specified by the triple <relation, predicate, value>, that is, by a relation, the relation's domain, and the relation's co-domain. An example triple would be <actor, bark, dog>. In fact, the actor relation is an important and salient relation in general because it answers the question “who did what?” The relation distinguishes, for example, explanations for why dogs bark from those explaining why seals bark. As seen in Figure 77, this index level is represented by placing a micro-index on the bark property of the actor symbol. A micro-index is simply a “gensym”<sup>101</sup> that is guaranteed to be unique. The micro-index is then given a dog property whose value points to the next level of indexing.

---

100. In the frame knowledge representation system, function abstractions exist to compute this mapping. The call (*\*FRAME\* frame-var*) retrieves a token's value, whereas (*frame-type frame-var*) returns the type of a given token.

101. Actually, the implementation uses a call to the LISP function *gensym* so that the symbol will be interned into the current symbol package.




---

Figure 77. Indexing for items about why dogs bark

A second level of indexing is specified by zero (or more) slots of the original relation's predicate. Each slot is represented by the tuple `<role value>`. For each slot, the index requires two micro-indexes. The first (i.e., `role`) represents the name of the slot. The second (i.e., `value`) represents one of three conditions. If the role-filler is an attribute value or literal, it represents the frame-value of the filler. If the role-filler is a relation, it represents the frame-type of the domain of the relation. Otherwise, it represents the frame-type of the role-filler. As an example, consider the object at which a dog barks. The filler of the `to` slot is the relation `at-location` whose domain is a container token. Thus, the index tuple is `<to container>`. See the level two portion of Figure 77.

The final level of indexing specifies the index type. That is, the index specifies the kind of memory at which it points. At the end of a sequence for an index, the retrieve or store (index) routine looks at or sets the memory-type. This level is represented with a final micro-index whose property is one of the following: either `question-type.0`, `xp-type.0`, `plan-type.0`, or `case-type.0`. The value of the property will be the element (or a list of elements) stored in memory (e.g., an XP for indexes of type `xp-type.0`).

The function `do-index` is the major memory storage function of Meta-AQUA. It takes a memory item to be indexed, the item's type classification, and a relation that serves as context for the mapping, and returns an instantiated index frame that represents the index. As a side-effect, it places the memory item in conceptual memory via the micro-indexing implementational scheme described above. Optionally, the memory item may be placed in conceptual memory along with any other structures that happen to be there, or it may overwrite what is already there, depending on the caller of the function. That is, destructive storage is optional.

In addition, every memory item added to conceptual memory through indexing is placed on a “retrieval list” for finding similar items when storing. Before a new item is indexed, the memory system performs a check on the retrieval list for the memory item's type to see if there already exists an item that is of this type. This action simulates a reminding at storage time so that the memory system can find forgotten or lost memories. This feature is necessary for **forgotten goal** and **missing association** errors. To make these features more concrete, the next section examines a specific example of opportunistic reminders. See also Section 8.5.4, “Forgetting a Learned Explanation,” starting on page 203.

### 8.4.3 Reminders in Opportunistic Memory

Consider the short story generated by Tale-Spin, TS2, as shown in Figure 78.<sup>102</sup> Given that Meta-AQUA believes that playing is constrained to children and that people hit or strike animate objects when they wish to hurt them, TS2 will generate a number of anom-



alies. When sentence S8 states that Dad plays with a ball, Meta-AQUA detects an anomaly because the token conflicts with its conceptual definition of playing. That is, it expects only children to be actors of playing events. Likewise, S9 conflicts with its knowledge of what kinds of objects people hit. People hitting inanimate objects cause Meta-AQUA to explain the action. The first anomaly will cause Meta-AQUA to reach an impasse because it does not have any explanation for why people play. The second anomaly will cause Meta-AQUA to hypothesize that Lynn wanted to hurt the ball. Neither of these anomalies can be resolved, so the program suspends them both in the BK. The first is indexed under adults who play with balls, whereas the second is indexed by children who hit animate objects.<sup>103</sup>

One day dad was bored. Dad asked Lynn, “Would you push the ball2 to me away from you?” Lynn went to the garage. She picked up the ball2. She had the ball2. She went to outside. He went to outside. (S8) He played with the ball2. (S9) She hit the ball2. (S10) She hit the ball2 because she wanted to move the ball2 to him. (S11) He hit the ball2. (S12) He hit the ball2 because he wanted to move the ball2 to her. (S13) He played with the ball2 because he didn't want to be bored.

--- The End ---

---

Figure 78. Tale-Spin story TS2

The story now supplies an explanation for why Lynn hit the ball in sentence S10. When this explanation arrives, it causes a reminding of the previous question “Why did Lynn hit the ball?” It finds this previous question in the BK and re-establishes it in the FK. The old question is located because the EXPLAINS node of the input XP (i.e., its consequent) is the hitting event that matches the index under which it was previously stored.<sup>104</sup> The explanation contradicts the expectation that Lynn wanted to hurt the ball, so as with the learning episode presented in Chapters VI and VII, Meta-AQUA is able to learn the new explanation, loosen its constraint on the objects at which people hit, and differentiate the two hitting explanations by re-indexing them with respect to each other.

---

102. Note the similarity between story TS2 and story HC1’ (“The handball game”) in Figure 69 on page 170.

103. The second index would be represented as actor→hit→child→to→ball→question-type.0→q, where q is the question be indexed.

104. The input XP is of type XP-GOAL-OF-OUTCOME->ACTOR which states that “if the expected outcome of an action results in a goal state for that agent, then the agent will choose to perform that act.” This is a basic assumption of rationality (Newell, 1982). See Ram (1989, 1994) for additional details concerning volitional XPs that explain why agents perform particular classes of actions.

As a result of this learning experience, when the program processes sentence S11, the concept is no longer anomalous. However, it is interesting to the system because it had recently learned about people hitting animate objects. This causes Meta-AQUA to explain the action. Instead of retrieving the old hurt explanation, the system applies the new explanation which is now indexed by `person-hit-toy-object`. The explanation is verified when S12 is encountered. Thus, not only does Meta-AQUA not repeat a failure, but it predicts the correct explanation before encountering it. Finally, when sentence S13 is processed, the system simply acquires the new explanation.

Story TS2 is significant for at least two reasons. For all three of the events (one play and two hit actions), the explanations for why the actors performed the action are given in the story, but they come at different points in the story. In particular, the explanation that answers why Dad plays with toy balls comes after the explanations for both hitting events even though the question "Why did Dad play with the ball?" was formed first. Opportunistic memory allows individual questions to be processed independently of the order in which the story provides new information. That is, opportunism allows interleaving of multiple hypothesis formations and verifications.

Secondly, this story represents a situation where Meta-AQUA actually benefits from learning within the same story in which learning occurs. There is no requirement that the entire story be processed before learning takes place. This incremental form of learning is more cognitively plausible than non-incremental learning systems (e.g., AUTOCLASS, Cheeseman, Kelly, Self, Stutz, Taylor, & Freeman, 1988) that process all input items before generalization or performing other forms of learning. The following section describes features of the learning system in more detail and provides an extended Tale-Spin example.

## 8.5 Learning Subsystem

The three chapters of Part Three, "A PROCESS THEORY OF LEARNING AND INTROSPECTION," have already described in detail the bulk of Meta-AQUA's learning mechanisms and algorithms. The two chapters of Part Two, "A CONTENT THEORY OF MENTAL REPRESENTATION," examined the major representations used by the learning system. This section looks somewhat closer at a few features not covered in this previous material. Section 8.5.1 reviews the basic architecture of the learning system and enumerates the available learning algorithms in the system's toolbox of learning methods. It also lists the IMXPs used by the system during learning. Section 8.5.2 discusses the space of failure causes that the implemented blame-assignment procedure considers when actually explaining a reasoning failure. Section 8.5.3 briefly discusses learning higher-order knowledge in the context of an Elvis World example. Finally, Section 8.5.4 works through the Tale-Spin version of example HC2, originally presented in Section 2.1.2.

### 8.5.1 Divisions of the Learner

The learning subsystem performs four functions. When the performance task fails, the learning system performs blame-assignment, decides what to learn, constructs a learning strategy, and then executes the strategy. The learner is partitioned into two parts. A case-based reasoning component performs the first two functions while a nonlinear planner performs the last two. As discussed in Chapter VI, the CBR module receives input in the form of a TMXP. The TMXP represents a trace of the failed reasoning detected during a previous performance task. During explanation of the prior reasoning (blame assignment), it retrieves an IMXP that helps locate causes of the failure and links the symptoms of failure with the faults. The IMXP also helps spawn a set of specific learning goals or changes to the system's BK (deciding what to learn). These goals are then passed to a non-linear planner, UM Nonlin v.1.2<sup>105</sup> (Ghosh et al, 1992) along with a predicate representation of the reasoning context. As discussed in Chapter VII, the planner creates a learning plan with which to achieve the learning goals (learning-strategy construction). To do this the learner uses schemas that are similar to the STRIPS planning-schemas designed for the Blocks World. The plan is executed by performing calls to particular learning algorithms specified by primitive actions in the learning plan (learning-strategy execution). Following the learning session, control is passed back to the performance system.

As mentioned, the learning system has access to a toolbox of learning algorithms from which the Nonlin component creates a learning plan. This toolbox includes a number of algorithms that were re-implemented so that they operate on a frame representation of conceptual entities used by the system. The algorithms currently contained in the toolbox are case-acquisition, explanation-based generalization (EBG), abstraction, and index learning. None of these algorithms perform the same task, so once the system identifies that a learning goal is necessary, it is unambiguous which method applies to the goal. That is, this research does not address the selective-superiority problem (Brodley, 1993). Instead, the research examines how to order and select learning methods at a coarse grain level in order to create a learning strategy that avoids learning-goal interactions. Future research will address the selective superiority problem and will incorporate more methods into the system's toolbox (See Section 10.1.3 on page 248).

The learning system has access to nine IMXPs that drive the learning process. The IMXPs are as follows:

---

105. Obtained from the University of Maryland at College Park at URL <ftp://cs.umd.edu/pub/nonlin> in file `nonlin-files.tar.Z`.

IMXP-SUCCESSFUL-PREDICTION  
 IMXP-EXPECTATION-FAILURE  
 IMXP-RETRIEVAL-FAILURE  
 IMXP-NOVEL-SITUATION  
 IMXP-NOVEL-SITUATION-ALTERNATIVE-REFUTED  
 IMXP-NOVEL-SITUATION-ALTERNATIVE-REFUTED-NO-ANOMALY  
 IMXP-BAFFLED-AND-RESOLVED  
 IMXP-ANOMALY-AND-BAFFLED  
 IMXP-ANOMALY-EXPLAINED

These representations are the most complicated knowledge structures in the program. To appreciate the complexity of these representations consider the representation for IMXP-BAFFLED-AND-RESOLVED in Figure 30 on page 89. Although the frame definition takes an entire page in eight point font, the figure is still incomplete. Some slots were removed in order to fit the page. Moreover, the IMXP is one of the more moderately sized IMXPs, not the largest.

The following section explains what failure-causes the system actually considers when making an assignment of blame.

### 8.5.2 The Implemented Space of Explanation Failures

As currently implemented, the blame-assignment phase of learning does not consider all of the failure causes enumerated in Table 5, “Detailed taxonomy of causes of reasoning failure,” on page 53. However, as mentioned in the introduction to this chapter, the implementation does consider many more of these causes than do most AI systems. At the present time, the system concentrates on errors that arise from missing and flawed domain information and the indexing of such information in the BK, that is, the “Knowledge States” columns of Table 5. Yet given this limitation, the combinations of failure encountered are many (see Figure 79), and, as will be explained in Section 8.5.3, “Learning about higher-order knowledge,” the resultant learning can be non-trivial.

Figure 79 graphically illustrates the space of failure causes that blame assignment considers in the experimental study presented in the next chapter. The shaded portion of the figure represents Meta-AQUA’s performance system when no failures are detected. The program first accepts a given input. If the input is anomalous, the system explains it, otherwise it checks to see if the input is in any other way interesting. If it is interesting, the system explains it; otherwise, it skims the input and accepts another. Outside of the shaded area represents the space of failures.

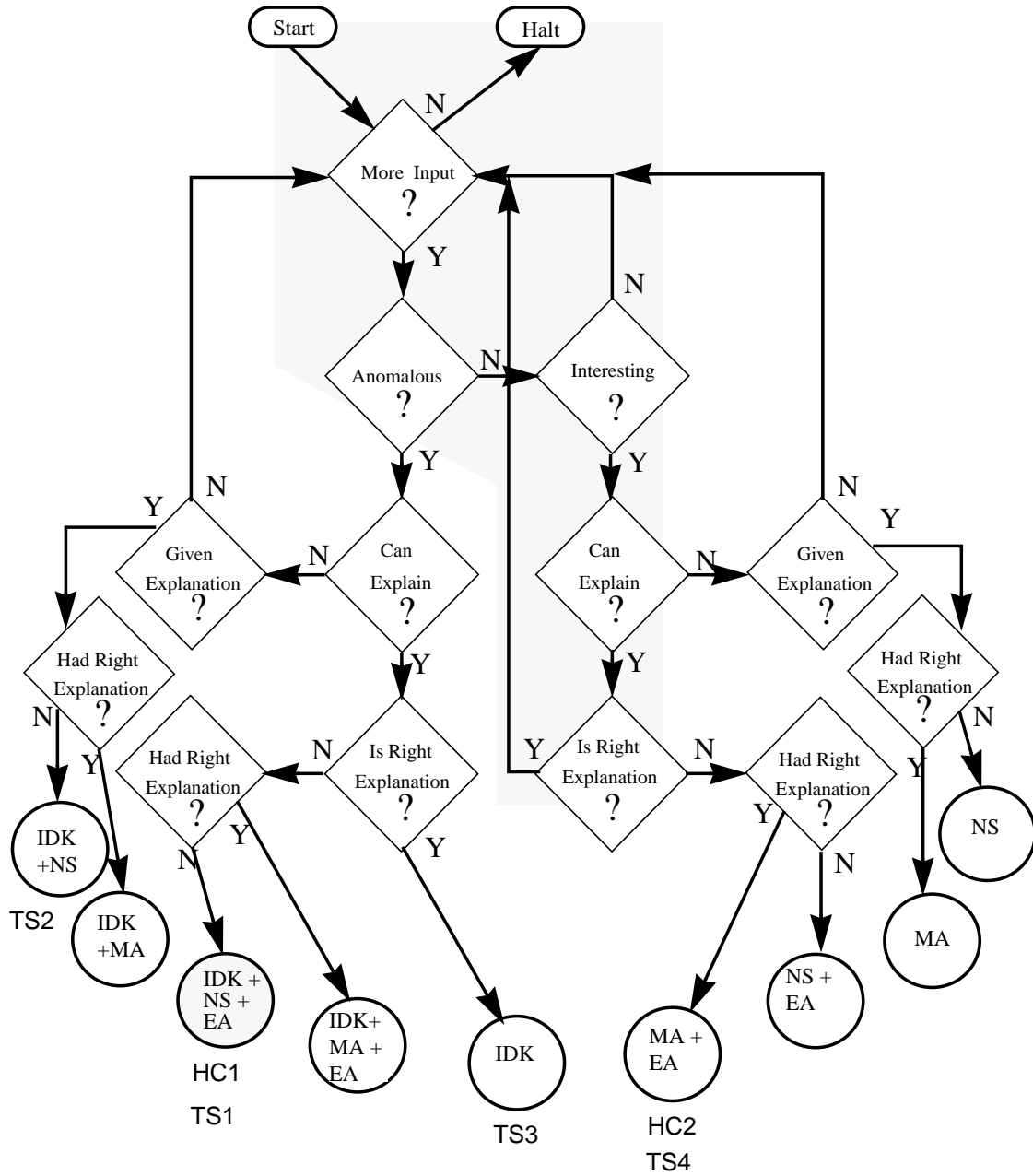


Figure 79. Implemented space of reasoning faults

IDK=incorrect domain knowledge; NS=novel situation;  
 MA=missing association; EA=erroneous association

When Meta-AQUA generates an explanation for an anomaly in the input story, the explanation may be incorrect. Alternatively, it may reach an impasse when trying to explain and thus not be able to generate an explanation at all. In the first case the explanation may be wrong, but the right explanation was in memory all along. If it cannot explain an anomaly, the explanation may have existed, but could not be found. All of these cases can occur both when the input is anomalous (left hand side of the figure) and when simply interesting (right hand side of the figure). An additional case occurs when Meta-AQUA explains an anomalous input correctly. It can then learn what was wrong with the knowledge such that it thought it was anomalous when actually it was not.

The circles at the bottom list the combinations of failure that occur given the situation (i.e., whether the explanation was given, etc.). The shaded circle represents the fault of the hand coded story HC1 as described in Chapters VI and VII. Annotations underneath the circles refer to stories described in previous or subsequent sections.

One may object that because these all map to a single fault, a decision tree could be built rather than going through the introspective process. However, this figure represents the conditions available only in hindsight or through the auspices of an oracle; it is a virtual flow-chart, not an actual flow of control in the program. Meta-AQUA must go through the blame-assignment process in order to determine the actual situation that applies to a given set of circumstances. For example, there is no way that the system can determine in advance that it has the right explanation in memory but failed to find it (i.e., has forgotten the explanation and so the error is *missing association*). A set of if-then statements will not suffice to perform blame assignment.

This set of failure combinations represents an exhaustive set of causes that can account for reasoning failure assuming correct processes, goals and input. Adding these other dimensions makes the blame-assignment task much more complex. However, we believe that deriving representations for these additional combinations will be tractable given a full analysis of the possibilities. Heuristics must developed, however, to estimate when the assignment of blame will be so difficult that metareasoning should not be pursued (see next chapter).

A number of other researchers have presented learning approaches to many of the other failure types not covered directly in the Meta-AQUA implementation, although they do not specifically consider the learning-strategy construction problem. Given this research, we can outline a number of relevant approaches to failure causes from Table 5 not covered in this implementation. Research from these areas can be used in the future to make Meta-AQUA more complete.

- *Input Noise*: The reasoner may possess the right knowledge, have it organized in a proper manner, and use the correct reasoning methods, yet fail due to incorrect or incomplete external knowledge sources. In reasoning tasks, the blame may be due to measurement errors, obsolete data, missing data, or explicit deception by another agent. The learning solution is to determine the conditions under which knowledge sources are reliable and the kinds of data that are necessary in a given situation (Booker, Goldberg & Holland, 1989). Data from human studies can be useful here to constrain the learning (e.g., see Johnson & Seifert, in press).

- *Incorrect Reasoning Choice*: This failure type occurs when the reasoner has an appropriate knowledge structure with which to reason and an index to the structure in memory, but incorrectly chooses the wrong knowledge because the reasoning method it decided to use turned out to be inappropriate or inapplicable. An analysis of the choice of reasoning methods results in learning control strategies designed to modify the heuristics (or add new heuristics) used in this choice (Mitchell et al., 1983; Sleeman, Langley & Mitchell, 1984).

- *Flawed behavior / Missing behavior*: The fault may occur because of incorrect procedural knowledge. Stroulia (1994) has presented an interesting metaphor that pertains directly to these failure causes. She treats a cognitive system as a device having reasoning components and models them with structure-behavior-function (SBF) models. This allows her Autognotic system to perform blame-assignment with the SBF models in the same manner that other systems diagnose physical devices in the real world. Such reflective diagnoses enables self-repair (learning).

### 8.5.3 Learning about higher-order knowledge

As described by Section 8.3.3, Tale-Spin outputs a CD representation of events in the story, and a translator converts these concepts into a frame representation that Meta-AQUA understands. Often the input from Tale-Spin does not match Meta-AQUA's conceptual definitions and so the system detects an anomaly (e.g., the input *dog-barks* concept is anomalous because the object at which the dog barks is animate whereas the conceptual definition from which the system compares the input constrains the object slot to inanimate). However, Meta-AQUA's conceptual knowledge for these primitive representations is not the only source of **incorrect domain knowledge** in the system. As explained in Section 8.2, during conceptual skimming a script application mechanism interprets the primitive acts given to it by using hierarchical knowledge from scripts. Therefore, the inferences generated by these knowledge structures themselves may also contain errors that lead to failure.

For instance, consider story TS3 in Figure 80. To represent sentence S26 ("Elvis smokes pot."), Tale-Spin generates the CD primitive **INGEST** whose *actor* is Elvis and whose *object* is marijuana.<sup>106</sup> This representation itself causes no anomaly because the

actors of **INGEST** are volitional agents and the objects may be plants. But, the script applicer incorporates the **INGEST** into a **Smoke-Pipe** scene of the **Smoking-Script**. It then examines the **Smoke-Pipe** scene for interesting input (and to see if it answers any previous questions). Now because **Smoke-Pipe** has a constraint limiting the ingredients of pipes to be tobacco, an anomaly will result because of the knowledge in the scene's definition, not the definition of **INGEST**.

One day Elvis was bored. Elvis pushed cupboard-door away from the cupboard1. The cupboard1 was open. He took the pipe1 from the cupboard1. He had the pipe1. The cupboard1 didn't have the pipe1. He pushed cupboard-door to the cupboard1. The cupboard1 wasn't open. He pushed fridge-door away from the fridge1. The fridge1 was open. He took the ganja1 from the fridge1. He had the ganja1. The fridge1 didn't have the ganja1. He pushed fridge-door to the fridge1. The fridge1 wasn't open. He poured the ganja1 into the pipe1. The pipe1 was filled with the ganja1. He took the lighter1 from the table2. He had the lighter1. The table2 didn't have the lighter1. He pushed the lighter1. The lighter1 was on. He moved the lighter1 to the ganja1. The ganja1 was burning. He pushed the lighter1. The lighter1 wasn't on. (S26) He smoked the ganja1. The pipe1 wasn't filled with the ganja1. The pipe1 was dirty. He exhaled the smoke1 into the air1. He pushed hot-faucet-handle away from the hot-faucet. The hot-faucet was flowing. He moved the pipe1 to the hot-faucet. The pipe1 wasn't dirty. He pushed hot-faucet-handle to the hot-faucet. The hot-faucet wasn't flowing. (S36) He smoked the ganja1 because he didn't want to be drug-withdrawing.

--- The End ---

---

Figure 80. Tale-Spin story TS3

Following the detection of the anomaly, the system then asks why Elvis smokes the marijuana. As a consequence, Meta-AQUA will use a tobacco smoking explanation to answer why Elvis chose to perform such an action. The resultant hypothesis is that the action of smoking the pot relieves the tension of withdrawal from the effects of an addictive substance. This explanation is later confirmed in the story (at S36), so the explanation is accepted. The subsequent learning uses an IMXP called **IMXP-ANOMALY-EXPLAINED** to change by abstraction the constraint of the smoking scene to be any plant, rather than just tobacco (unfortunately an overgeneralization).

So although only one failure type was involved (**incorporation failure**) and only one learning algorithm needed (abstraction), even simple learning can be complex. The problem is complex because inference is involved rather than just matches against concepts in the BK.

---

106. For a full trace of the system behavior under this example, see Appendix B.



### 8.5.4 Forgetting a Learned Explanation

Even when a system learns new concepts or changes old ones, the successful use of this information may not always occur if the BK is not organized to promote the retrieval of it when needed. The examples of Section 2.1 illustrated that, even though Meta-AQUA may learn a new explanation in one story, it can fail to reuse the explanation in a subsequent story. A retrieval impasse can occur because, given the cues that compose a probe into memory, the existing explanation can lack the proper index with which to traverse the BK and thus to find the item. In effect, Meta-AQUA can forget learned explanations and can expect instead to acquire this knowledge anew. When Meta-AQUA is given the correct old explanation, the system must be reminded of the explanation that already exists in memory. Therefore, rather than pursuing a goal to acquire and expand the knowledge, it must be able to switch to a goal of reorganizing the knowledge. That is, learning goals are not static; even a system that uses the introspective method of deciding what to learn must be prepared to change its learning goals dynamically.

Consider the earlier scenario examined in Chapters VI and VII. The hand-coded story, HC1 (also the automatically generated story, TS1), represents a particular configuration of events. In the story, a police dog barks at some luggage in an airport (or at a vase in Elvis' home in TS1). Meta-AQUA considers the event to be anomalous because the system believes that dogs bark only at animate objects. As was seen earlier, the program eventually learns that dogs can bark at any physical object, including inanimate ones. It also learns the new explanation that "dogs bark when detecting contraband." After processing HC1 (TS1), Meta-AQUA's memory contains knowledge of two explanations for why dogs bark: an explanation for dogs that bark because they are threatened (indexed by `dog-barks-at-animate-object`) as well as an explanation for dogs that bark because they detect contraband (indexed by `dog-barks-at-container`).<sup>107</sup>

Tale-Spin then generates story TS4 (see Figure 81) and outputs it to Meta-AQUA. In this story, Elvis and Lynn are about to play with the ball when the police arrive at the house with a canine unit. The dog immediately goes to a throw rug and sniffs at the object (S23). When the dog barks (S24) the officer pulls back the rug to find Elvis' stash of marijuana. Consequently, the officer arrests Elvis (S30) and takes him away (S32, S33). The story then informs the reader that the dog barked because it detected the contraband (S35). Because Elvis loses his freedom due to the arrest, he can no longer play ball with Lynn, and so he remains bored (the original problem that motivated the story).

---

107. See Section 8.4 for the representation of these two indexes, especially Figure 77.

One day Elvis was bored. Elvis asked Lynn, "Would you push the ball1 to me away from you?" Police-and-dogs arrived. Officer1 went to outside. The police-dog1 went to outside. He pushed door-bell-switch1. The door-bell1 was ringing. He didn't push door-bell-switch1. The door-bell1 wasn't ringing. The phone1 was ringing. Mom picked up phone-receiver1. The phone1 wasn't ringing. She had phone-receiver1. She let go of phone-receiver1. She didn't have phone-receiver1. The cat1 pushed the vase2 to the floor1. The vase2 was broken. She pushed light-switch1. The light1 was on. He went to the kitchen. The police-dog1 went to the kitchen. The police-dog1 went to the rug1. (S23) The police-dog1 sniffed the rug1. (S24) The police-dog1 barked at the rug1. The police-dog1 was barking. He went to the rug1. He took the ganja1 from the rug1. He had the ganja1. The rug1 didn't have the ganja1. (S30) He arrested Elvis. He controlled Elvis. (S32) He went to outside. (S33) Elvis went to outside. The police-dog1 went to outside. (S35) If the police-dog1 detects the ganja1 then the police-dog1 will bark at the rug1. Lynn went to the garage. She picked up the ball1. She had the ball1. She went to outside. He asked her, "Would you push the ball3 to me away from you?" He asked her, "Would you push the ball2 to me away from you?" He was still bored.

--- The End ---

---

Figure 81. Tale-Spin story TS4

Immediately after the dog barks at the carpet, Meta-AQUA generates a question to explain why the dog barked (see figure Figure 82). The reason for this decision is that the system has recently learned about dogs and barking, so it is interested in any subsequent information that may be related. However, because the dog is barking at a rug and such an object is not a container, it does not retrieve the newly learned detection explanation. The dog also is not barking at an animate object, so the old threaten explanation is not retrieved. Instead, it can generate no explanation to explain the interesting story concept.

Reviewing the reasoning trace that preceded the conclusion, Meta-AQUA characterizes itself as “baffled” (impasse during memory retrieval). The system retrieves an IMXP based on this characterization, which helps it explain its own reasoning failure. The structure is unified with the representation of the original reasoning (stored in a TMXP) which produces the instantiation partially shown in Figure 83, “Instantiated forgotten detection explanation.” The knowledge structure shows that memory retrieval produced no explanation in response to the system’s question. Instead, a later input produced the answer.

The IMXP suggests that a knowledge-expansion goal be spawned to generalize the input explanation. This suggestion comes from the **potential-learning-goal** slot of the IMXP (see Figure 30., “IMXP frame definition for forgetting” on page 89). Conditions attached to the knowledge-expansion goal allow it to be posted if the node A was either acquired from the story or inferred, but not if it was retrieved from memory. A knowledge-organization goal is also spawned in order to index the generalized explanation in memory. These goals can be achieved by performing explanation-based generalization (EBG) on the

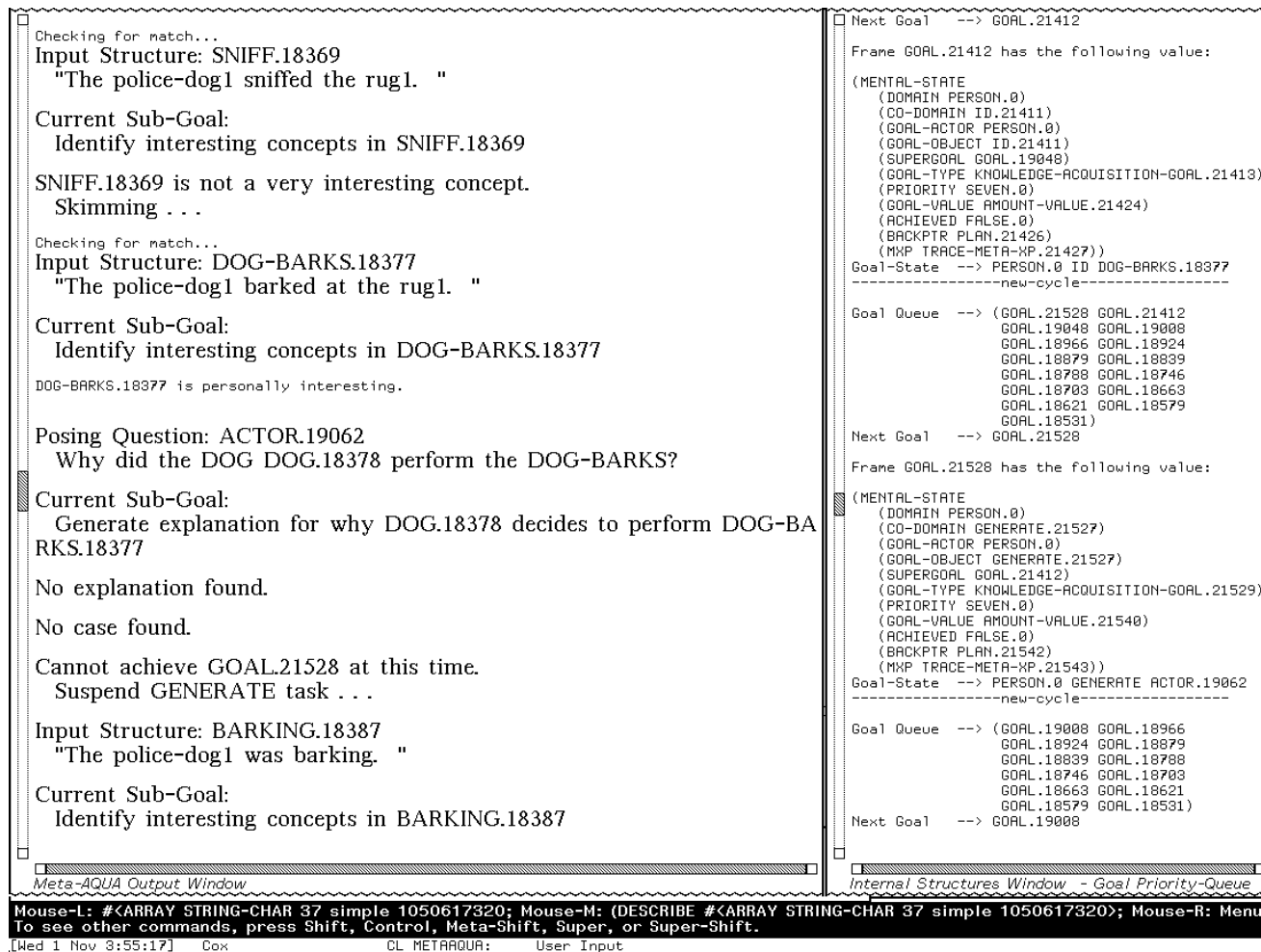


Figure 82. Meta-AQUA output during hypothesis generation phase of TS4

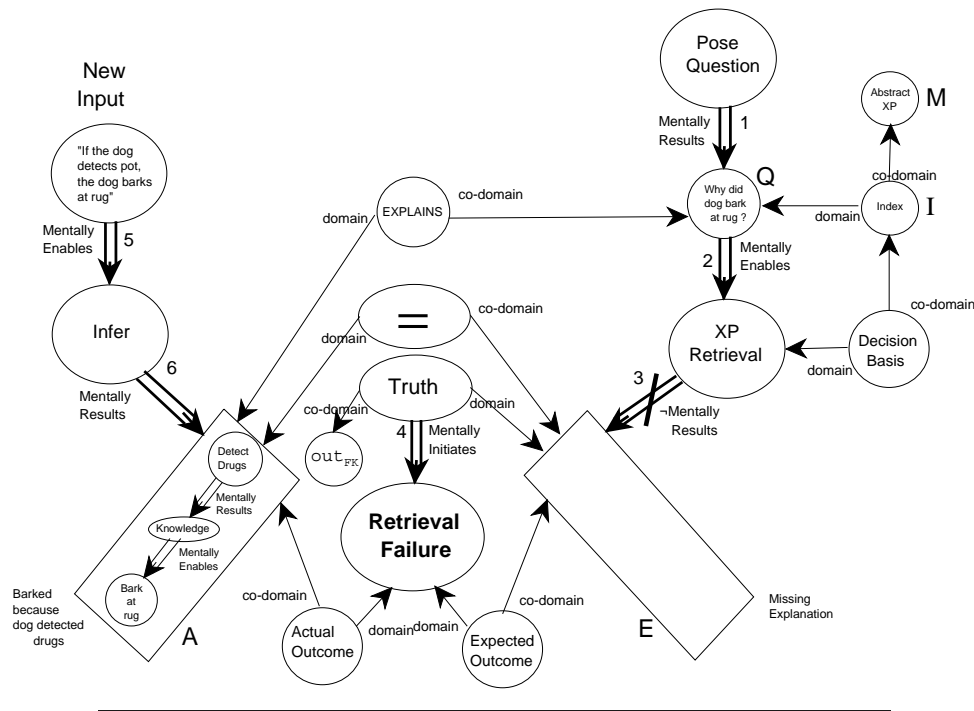


Figure 83. Instantiated forgotten detection explanation

new explanation (node A) and then indexing the explanation by the context in which the system encountered the explanation.

The system cannot determine *a priori* whether an abstract XP (node M) actually exists in memory but could not be recalled (thus, the failure cause is a missing association, I), or whether the system lacks the knowledge to produce the explanation (thus, the cause is that the situation is novel, i.e., M is missing). The system thus poses a question about its own IMXP (cf. Oehlmann, Edwards & Sleeman, 1994), “Does M exist in memory?” If M is missing, I is also missing; thus, the right question to ask is whether M exists, not I.<sup>108</sup>

The answer to the introspective question is obtained by performing EBG and then watching for a similar explanation in memory when it stores the new explanation via the indexing algorithm. As briefly described in Section 8.4.2 (specifically see page 194), the system can detect the presence of similar memories by maintaining a list of pointers to memory items for each conceptual type. At storage time, Meta-AQUA traverses the list,

108. Note that it cannot be the case that I is erroneous. If it were true, then some explanation would have been retrieved, although it may have been inappropriate.

checking each to see if it can unify the new memory with any of the older ones.<sup>109</sup> Meta-AQUA thus finds the explanation produced by the previous story (see Figure 84).

Merging the two explanations produces a better explanation: Dogs may bark at objects that hide contraband, not just at containers that hold contraband. The algorithm that indexes the generalization searches for the common ancestor of the object slots of both explanations; that is, objects that contain other objects and objects that cover other objects. This common ancestor is the type `hiding-place`. Thus, so that these types of explanations will not be forgotten again, the system indexes the explanation by “dogs that bark at potential hiding places” and places a pointer to the merged explanation on the memory list for the symbol `causal-relation`.

As a result of its learning, Meta-AQUA not only detects no anomalies in stories such as HC3 (Figure 85), it predicts the correct explanation in S7 while processing S5. Most importantly, however, this story illustrates the fact that learning goals are not static, but rather, that they are subject to dynamic re-evaluation, even when the planner that creates a learning plan knows about interactions. Some facets that bear on the pursuit of learning goals cannot always be anticipated in advance. In this case, the system had decided that it needs to acquire a new piece of knowledge, but instead it discovers that it had the knowledge all along. So instead of achieving a knowledge expansion goal to generalize and store what it thinks is a new explanation, it rediscovers the old one and changes the learning goal to a knowledge organization goal.

- S1: A person was outside a house.
- S2: The policeman approached the suspect.
- S3: His dog followed.
- S4: The policeman saw that the person was near a compost pile.
- S5: The dog barked at the compost pile.
- S6: The authorities arrested the suspect for drug possession.
- S7: The dog barked because he detected drugs.

---

Figure 85. Subsequent test story (HC3)

---



---

109. This mechanism simulates a memory such as that of DMAP (Martin, 1989, 1990), whereby memory items map to areas that contain similar memories. Although Meta-AQUA's mechanism is only a crude approximation to such architectures, the emphasis of IML theory is on the reasoning about memory (or other reasoning processes), rather than on a representation of the memory architecture *per se*. A more realistic mechanism would be for Meta-AQUA to use the generalized XP as a probe to memory to see if it is now reminded of the old XP. The current method suffers from the fact that it always finds the old XP at an unacceptable search cost.

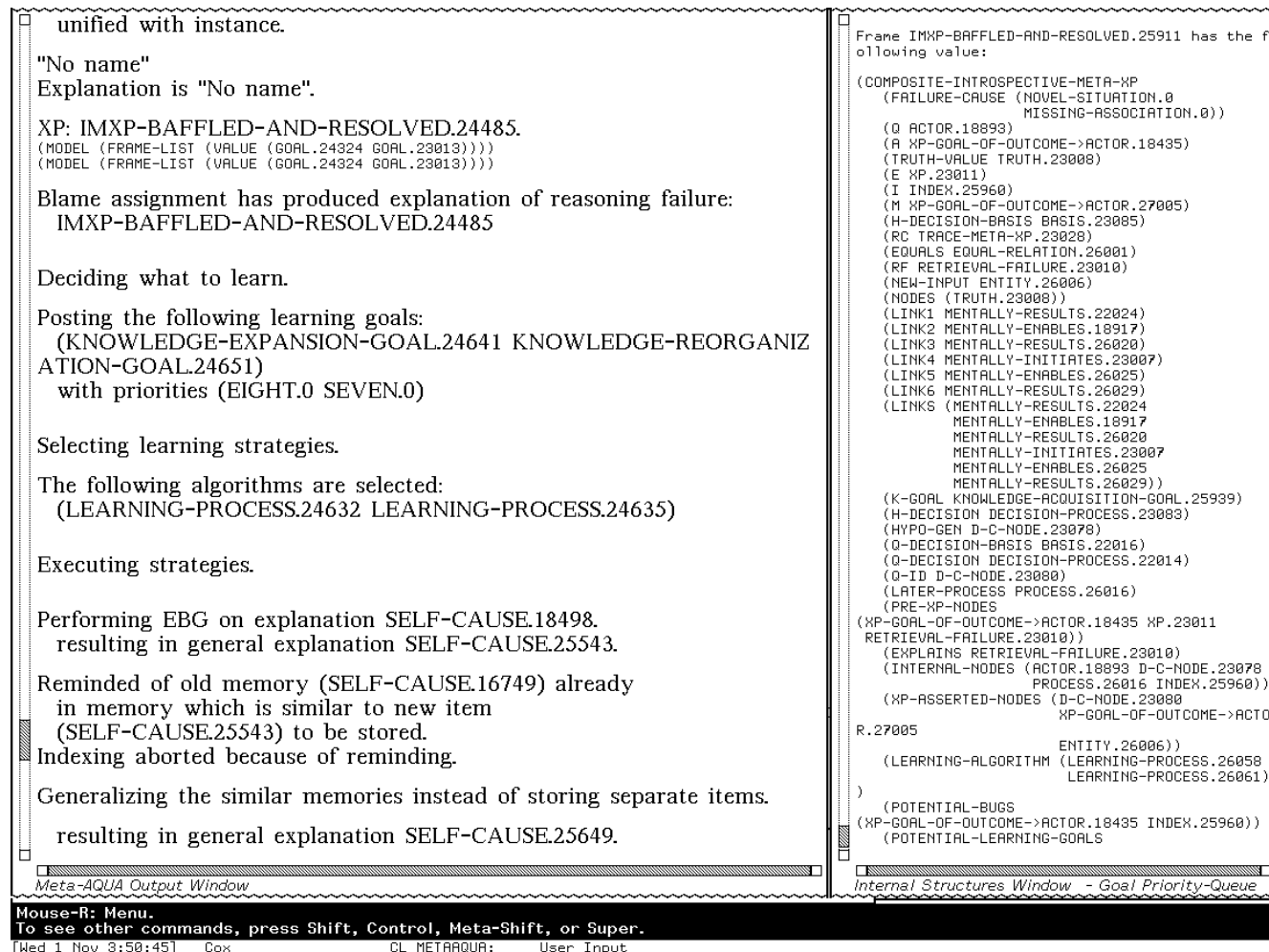


Figure 84. Learning phase during TS4

## 8.6 Summary

This chapter described the Meta-AQUA implementation including the four major subsystems (not including the frame representation system). These are the performance subsystem, the input subsystem, memory, and the learning subsystem. The performance system is story understanding. Among new implementational details this chapter discussed concerning the performance system was the script application module and its implication for learning about higher-order knowledge. The input system not only provides hand-tailored stories as seen in previous chapters, but, as introduced in this chapter, it also includes an automatic story generator called Tale-Spin. The memory system is an indexed memory divided into the BK and FK. An important feature of the memory is that opportunistic reminders are supported so that multiple hypothesis formations and verifications can be interleaved within a series of inputs. We also saw that Meta-AQUA can benefit from learning in a particular story before the story is even finished.

This chapter also reintroduced the learning subsystem. When discussing the learning system, an example of forgetting illustrated a number of points. First, forgetting can occur when the indexes for items are poorly organized in memory. Secondly, and more importantly, even if it knows about interactions between learning methods, the learning system must be prepared to change dynamically the learning goals being pursued. Finally, the example demonstrated the utility of introspective questions.

Markovitch and Scott (1993) characterize learning systems in terms of filters placed in an information flow through a system. Meta-AQUA possesses an input bias at the front end in the information flow; that is, the bias is to prefer failed experience. Markovitch and Scott call such a filter *selective experience*. They divide selective experience into three types: error-based, uncertainty-based, and miscellaneous heuristics. The examples presented in this thesis are all error-based, although the scope of the selective-experience filter in Meta-AQUA goes beyond their formulation because, as explained in Chapter III, error has numerous variations, only one of which (contradiction) Markovitch and Scott consider. Moreover, they claim that error-based filters are useful only when the input is in the form of problem/solution tuples. During the impasse of story TS4 (Section 8.5.4), however, Meta-AQUA generates no solution, yet the system was still able to learn a valuable lesson from the experience.

Meta-AQUA filters input examples in a relatively passive manner. It waits for failures to occur, then processes them by explaining the failure, deciding what to learn, and constructing a learning strategy. Another issue to pursue would be to have the system try to actively generate failed experiences in order to test or disprove some hypothesis or to generate learning experiences for some performance task. As Appendix A asserts, however, the system must be sensitive to inductive policies concerning the task domain. Currently,

the ability of a system to actively challenge itself and its knowledge is beyond the scope of our research.

With the implementational details presented by this chapter in hand, the next chapter can now examine how the total system is evaluated in an empirical study of the benefits of introspective multistrategy learning.



## CHAPTER IX

### EVALUATION

*The answer is that we test the validity of an empirical hypothesis by seeing whether it actually fulfills the function which it is designed to fulfil. And we have seen that the function of an empirical hypothesis is to enable us to anticipate experience. Accordingly, if an observation to which a given proposition is relevant conforms to our expectations, the truth of that proposition is confirmed. One cannot say that the proposition has been proven absolutely valid, because it is still possible that a future observation will discredit it. But one can say that its probability has been increased.*

—Alfred Jules Ayer (1936), p. 99.

Unlike most performance tasks in machine learning where optimal or provably correct solutions to well defined tasks exist (e.g., classification, path planning, and puzzle solving), it is unclear how to empirically evaluate understanding tasks fully. The question of what it means for an agent to comprehend a story does not have a simple answer in objective terms. Solutions to comprehension tasks are often qualitative, in contrast to quantitative tasks such as mathematical problem-solving. Within mathematics a definitive answer to the question, “What is  $2 + 2$ ?” exists. However, to accurately answer the question “Why did Elvis smoke marijuana?” requires inference and subjective interpretation (Carbonell, 1981) with respect to Elvis’ goals and values. Moreover, the context of the question may be relevant for understanding why the question is being asked and, therefore, for determining the best explanation; whereas, the context is irrelevant to answering the addition question.<sup>110</sup> For instance, the most relevant answer may be “Because Elvis is on a diet” if, prior to the question, the reasoner is told that Lynn just baked pot brownies. No absolutely correct answer exists. Rather, many answers at different levels of specification can suffice depending upon the context, the question, and the knowledge of the reasoner (Ram & Leake, 1991). One contribution of this research is to provide a quantitative measure that de-emphasizes complete and consistent explanations. Instead, it assigns “partial credit” when evaluating how pro-

---

110. The context matters only if the immediately prior addition problems have been performed in base three. In this case the answer would be 11 rather than 4.

grams understand and explain the input they experience. But more important is to evaluate how an understanding system learns, that is, how it improves its ability to understand.

The Meta-AQUA implementation tests the theory of introspective multistrategy learning presented in the previous chapters by providing a specific commitment to the concepts within the theory. But the assertion that the implementation tests the model immediately raises the questions “In what way can it be considered a test, and what are these commitments?” That is, how does the program validate the theory, and how can it be said to be falsifiable? The implementation effort by itself is falsifiable in a sense. If the theory cannot be implemented by the researcher, then the theory can be said to be underspecified. But although the last chapter showed that the theory passed this basic falsifiability test, this chapter presents a more rigorous hypothesis, complete with a null hypothesis that can falsify the theory if accepted (see Section 9.1.1.2).

However, the theory is difficult to validate completely and with confidence. It is not a strict psychological theory designed to explain a particular set of known human data, nor is it a computational theory embedded within a conventional machine learning paradigm having an agreed-upon evaluation criterium. A cognitive psychologist cannot simply run a couple of human experiments that prove or disprove its validity conclusively. The computer scientist cannot simply run its learning algorithm against a standard test suite (e.g., domain theories from the UC Irvine Machine Learning Repository) in order to compare it to other learning algorithms, nor can the theoretician derive an average-case mathematical model to predict the algorithm’s accuracy and then compare it to actual behavior (e.g., Pazzani & Sarrett, 1990). Rather, the phenomena of interest (introspections and their role in learning) are at a large grain size; the performance task (story understanding) is subjective; the hypothesis (introspection facilitates learning) is controversial. Such a stance places the researcher in the position of having no firm conventions upon which to rely when evaluating the theory; however, the advantage of it is that significant molar behaviors can be examined and interesting issues entertained that fall outside of the standard paradigm (Schoenfeld, 1992).

So why do cognitive scientists build programs? Many researchers (e.g., Newell & Simon, 1963; Simon, 1995) consider AI programs to be theories. But to equate the theory with the implementation is not necessarily warranted. The theory is contained in the representational formalism and the abstract algorithms, not in the program within which these concepts are embodied. The programs are the means for carrying out empirical experiments on the theory, much like psychological experiments test ideas contained in theoretical assertions about the mind or about behavior. The programmed configuration of processes, knowledge, and input and the resultant program output can themselves be the objects of study and comparison with human behavior that lead to greater confidence (or doubt) in a given cognitive theory (Simon & Halford, 1995). But unlike psychological experiments, computer simulations do have the additional burden of separating the imple-

mentational details needed to make a program run in a given language from the theoretical claims upon which the program is designed to test.

The instantiation of theories into programs has benefits as well as burdens. The process of implementing a theory in a program forces the theorist to commit to specific computational mechanisms, such as the assertion that learning entails blame assignment, decisions to learn, and strategy construction. A computational commitment also reveals where the theory requires adjustment. For example, an early assumption of IML theory was that Meta-XPs possessed uniform representation (Cox, 1991). As the implementation developed, however, it became apparent that the computation needed two types of meta-explanations: one to explain *how* the reasoning fails (TMXP) and another to explain *why* the reasoning fails (IMXP). The function of each type is different. So, the evolution of the theory and its commitments is facilitated by the constructive feedback provided by machine output. This chapter presents an evaluation of the current manifestation of these commitments.

We provide a detailed description of how IML theory is evaluated, both from a computational and a psychological perspective. Section 9.1 briefly presents two hypotheses stemming from this research and outlines how they are tested. Section 9.2 provides a computational evaluation of the first hypothesis using data generated by the Meta-AQUA program with automatically generated input, while Section 9.3 evaluates the second hypothesis by modifying Meta-AQUA to simulate psychological protocols of human subjects. The final section, Section 9.4, summarizes the results and discusses additional issues.

## 9.1 The Hypotheses

The first and last sentences of the opening paragraph of the thesis summary (p. xxix) form the two testable hypotheses of this research:

*The thesis put forth by this dissertation is that introspective analyses facilitate the construction of learning strategies.... Thus, the object of this research is to develop both a content theory and a process theory of introspective multistrategy learning and to establish the conditions under which such an approach is fruitful.*

In general terms, the hypotheses and claims of this thesis are as follows:

### ***Hypothesis 1. Introspection facilitates learning***

More specifically and as operationalized by Section 9.1.1.1, this statement is equivalent to the assertion that the rate of improvement in story understanding with learning goals exceeds that of story understanding without. The results from a test of this assertion will show that a loose coupling of blame-assignment and repair is preferred to a tight coupling. That is, the deciding to learn stage is necessary for effective learning.

***Hypothesis 2. IML theory is a sufficient model of introspection***

As Section 9.1.2 makes operational, this statement seeks to test whether the implementation of IML theory as embodied in the Meta-AQUA program is cognitively plausible. It passes the test if, after making a small number of modifications, the program is sufficient to cover a set of human data concerning the relationship between learning and metacognition. As will be argued in Section 9.3.3, additional support for this hypothesis comes from the theory's generality. That is, it can account for introspective learning in two sets of humans protocols in two separate domains (learning to program in LISP and learning to troubleshoot electronic circuits), and in one set of artificial data in yet another domain (learning to understand stories).

Although much future research remains to determine a full evaluation of these assertions, this chapter presents a number of results that directly support them. In essence, the approach to examining the two statements above is as follows. First, holding all other factors constant in Meta-AQUA, if the rate of improvement in the performance task (i.e., the learning curve) is greater with introspection than without, introspection must be responsible for the improved performance. Second, if, with minimal changes, the Meta-AQUA model can cover real human data on metacognition and learning, then the theory is a sufficient one.

### **9.1.1 Hypothesis Number 1: Introspection facilitates learning**

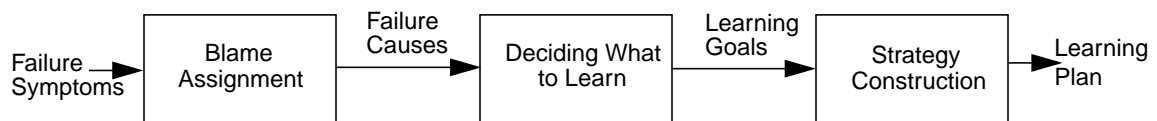
One approach to establishing the claims set forth in the previous section is to perform a kind of ablation study. Surgically removing the learning goals eliminates part of the system's mechanism responsible for introspection. The intention of this manipulation is to show different empirical learning curves with and without introspection as a function of the number of inputs.

The methodology below not only tests the hypothesis that introspection facilitates learning, but also it more directly supports the position that a loose coupling of blame-assignment and repair (via learning goals) is preferred to a tight coupling approach.<sup>111</sup> But perhaps more importantly, this methodological approach also subjects to scrutiny the claim that the second phase of learning, deciding what to learn, is *necessary* for effective learning.

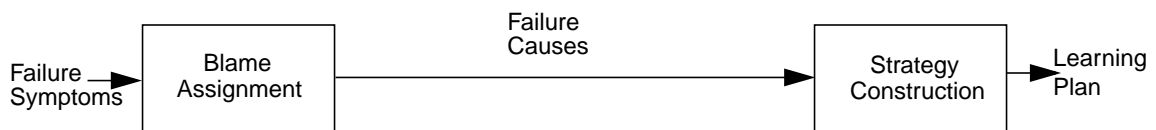
IML theory is the only learning theory that makes such a strong claim. Few computational systems other than Meta-AQUA include an explicit calculation of a goal to learn and then use that goal to influence learning. So in addition to the arguments and hand-coded examples from Chapter VII that support this position, this chapter presents quantitative evidence that supports the utility of this stage.

### 9.1.1.1 Fully introspective and semi-introspective learning

As presented in this document, introspective learning is a computational process with a decomposition as shown in the upper portion of Figure 86. *Fully introspective multistrategy learning* consists of examining one's own reasoning to explain where the reasoning fails. It consists further of knowing enough about the self and one's own knowledge that the reasoner can use such meta-explanations when deciding what needs to be learned. As reiterated throughout this document, introspection amounts to performing blame assignment and subsequently posting explicit goals to learn. Learning amounts to the construction of a learning plan designed to change the reasoner's knowledge and thereby to achieve the learning goals.



Three phases of fully-introspective multistrategy learning



Two phases of semi-introspective multistrategy learning

Figure 86. Learning goal ablation

111. See Section 7.1, “Blame Assignment and Repair: Tight versus loose coupling,” starting on page 212 for a detailed discussion of the issue.

One measure with which to test the hypothesis that introspection facilitates learning is to remove the goals from the introspective process above, leaving a more reflexive activity. That is, while holding all other variables constant, the system can be run with and without learning goals to determine the effect of introspection. Instead of using the explanation of failure created during the blame-assignment phase to post a set of learning goals that then direct the construction of a learning plan, the explanation can directly determine the choice of repair methods as shown in the lower portion of Figure 86. This short-circuited method is called *semi-introspective multistrategy learning*.<sup>112</sup>

Because the most meaningful comparison of the two types of learning is between their respective learning rates, rather than their relative system performance, the evaluation of each must be with respect to a base performance of the system with no learning. This requirement entails that three experimental conditions represent the independent variable. Holding steady all program parameters, the input, and the initial program-state, the first experimental condition includes learning goals, a second removes learning goals, and the third removes learning altogether. Section 9.2.1 on page 219 presents further details concerning the exact method by which learning is performed without learning goals.

### 9.1.1.2 The null hypothesis

The test of hypothesis number one includes a null hypothesis. If the learning rate with learning goals does not significantly improve the performance of the system compared to the learning rate without learning goals, then introspection does *not* facilitate learning. A possibility arises, however, that the bulk of the power of the fully introspective method actually resides in the blame-assignment side of the IML process depicted in the upper half of Figure 86. Blame-assignment is also introspective and has a reflective component. Perhaps Meta-XP's, rather than learning goals, provide the functionality responsible for effective changes in the BK. In any event, to determine fully the relative contribution to learning, both parts of the IML process (i.e., the case-based introspection side and the non-linear planning side) should be examined.

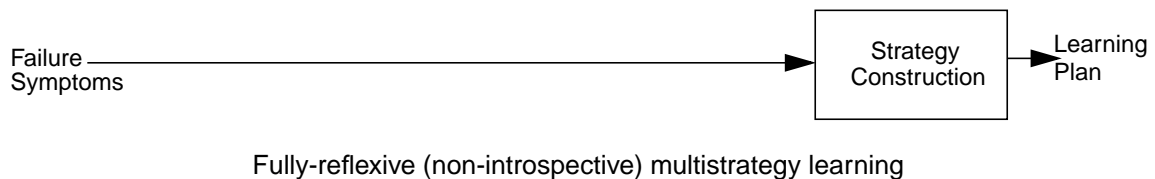
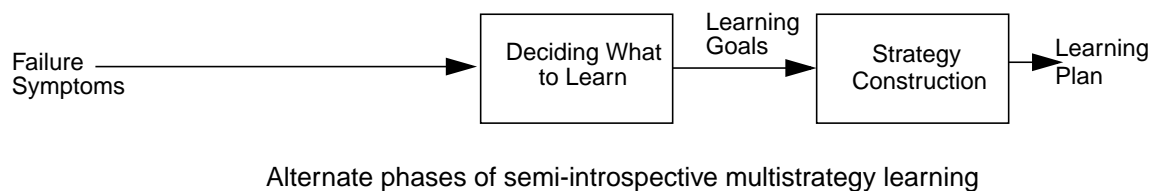
So although left for future research, to determine more completely the actual aspects of introspection that are most responsible for any utility gained during learning (and to safeguard the null hypothesis), the experiment should also perform the following two manipulations. First, by removing blame-assignment, the system maps failure symptoms directly to learning goals (see upper half of Figure 87). Secondly, to insure that an interaction between the two learning phases is not responsible for the effects, the experiment should

---

112. It is semi-introspective because, although part of the introspective process has been removed, the introspective mechanics of blame-assignment remain. The next section will discuss the ramifications of this further.

include a fully-reflexive method. That is, the system maps failure symptoms directly to learning algorithms, hence *non-introspective multistrategy learning* (lower half of Figure 87). This chapter presents the first of the three potential manipulations (i.e., removing learning goals). Future research will investigate the other two.

Given negative learning interactions such as those demonstrated in Section 7.2, the experimental expectation is that fully introspective multistrategy learning will outperform semi-introspective multistrategy learning. Section 9.2 provides the experimental design, the data, and the results that bear on hypothesis number one.




---

Figure 87. Alternate ablations

### 9.1.2 Hypothesis Number 2: IML theory is a sufficient model of introspection

Previous chapters describe a model of introspective learning that constructs a learning strategy at a micro-level. The resultant learning plan consists of calls to machine learning functions from standard inventories and so appears to be an engineering approach to learning. In such light, the theory is a response to a computer science problem concerning the proper integration of machine learning algorithms in complex situations. And although the framework within which this introspective process operates is cognitively oriented (e.g., the theory emphasizes memory access rather than search), it is perhaps asking too much for the reader to accept the implementation as a theory of human introspection. Nonetheless, many aspects of the theory pertain to human behavior and cognition in the real world (e.g., the taxonomy of failure applies as equally to human reasoning failure as it does to logical errors of computation).

Despite the technical orientation, a useful cognitive science experiment is to examine whether the theory presented here could model a macro-level learning behavior. Of interest are the molar learning strategies exhibited by human novices as they acquire complex cognitive skills and domain knowledge. That is, we are interested in better understanding goal-driven learning at the level of deliberate behavior. This entails an approach where the learning goals are not micro-goals, such as the goal to reconcile an input with a conceptual category (at best this kind of a goal represents some lower level generalization process), but instead the learning goals represent explicit desires for knowledge change in the face of real-world learning problems.

Recker and Pirolli (1995) have shown that a Soar-based model of learning called SURF can explain individual differences exhibited by human subjects while learning to program in LISP using instructional text. The difference that accounted for much of the variability was self-explanation strategies. Those students who explained problems to themselves during comprehension of the instructions performed well on a subsequent performance task consisting of LISP programming exercises. The students who did not exhibit this behavior were not as likely to excel in the LISP task. The SURF model predicted such differences. The model took into account only domain-related elaborations; however, subjects exhibited other self-explanations that the model did not cover. In particular, some subjects seemed to exploit metacognitive feedback, like comprehension monitoring, in order to judge when to learn (Pirolli & Recker, 1994). If self-reflection on the states of a subject's comprehension of the instruction indicated an understanding failure, then this was sometimes used as a basis to form a learning goal.

These data are well-suited for implementation in the Meta-AQUA framework and have the virtue of already existing. An additional benefit is that both the data and the Meta-AQUA model arose independently. If Meta-AQUA can be changed with minimal effort, including the addition of the learning strategies suggested above, then there is evidence that IML theory is a reasonable model of introspection. In addition, the model gains credibility if it can be extended to a new performance domain: instructional text comprehension in addition to story understanding. If the changes to the model are significant, however, then the believability of our theory as an approximate model of human metacognition is lessened. Section 9.3 reports the outcome of this experiment.

Q0: How can IML theory be evaluated?

Ans0: Test Hypotheses 1 and 2.



## 9.2 Computational Evaluation

This section presents the results of computational studies performed with Meta-AQUA to test the first hypothesis. Section 9.2.1 describes the independent variable used in this experiment. The program is run with learning goals, without learning goals, and with no learning in order to compare learning by direct mapping of repairs to learning arbitrated by goals. Section 9.2.2 describes the dependent variable (i.e., the measure chosen to evaluate learning). This measure quantifies a kind of “partial credit” for self-explanation by assigning points for generating questions, additional points for issuing any answer, and still more points for computing the correct answer. Next, Section 9.2.3 reports the data collected in the computational study. It establishes the claim that introspection facilitates learning by showing that the learning curve of fully introspective multistrategy learning indicates an improvement over the learning curve of semi-introspective multistrategy learning.

### 9.2.1 The Experimental Conditions (Independent Variable)

As discussed in Section 9.1.1, learning rates relative to a baseline no-learning condition are compared between a fully introspective and a semi-introspective version of Meta-AQUA. The independent variable that effects this change is the presence and influence of learning goals. The first experimental condition is called the learning goal (LG) condition and represents Meta-AQUA as described in Chapter VIII. The LG condition builds a learning strategy guided by the learning goals spawned by the IMXP that explained the failure and hence represents a loose coupling approach between fault (failure cause) and repair.

The second condition is called the random learning (RL) condition. Given the explanation of the causes of failure the system can directly assign calls to particular learning algorithms for each fault. The construction of the learning plan is then performed by a random ordering of these function calls, rather than by non-linear planning to achieve the learning goals. Without the learning goals, the RL condition represents a tight coupling approach, that is, a direct mapping from fault to repair.

The final condition is called the no learning (NL) condition in which Meta-AQUA performs story understanding, but if a failure exists, the system constructs no learning strategy. This condition represents the base line performance from which both the LG and RL conditions can be compared. Surprisingly, conditions exists for which no learning outperforms random learning (see Section 9.2.3.2 on page 224).

Holding all variables constant except the independent variables, Meta-AQUA is given input from the Tale-Spin problem generator and a dependent variable is measured. The

next section describes the criteria by which the performance of the system is measured, thus determining the dependent variable.

### 9.2.2 The Evaluation Criteria (Dependent Variables)

In previous research, paraphrase and question answering tasks have been used to measure successful story understanding (e.g., Cullingford; 1978; Lehnert, Dyer, Johnson, Yang, & Harley, 1983; Wilensky, 1978; Schank & Riesbeck, 1981).<sup>113</sup> If a reader sufficiently understands a body of text, then the reader should be able to summarize the central points of the story and list the major events within the story. If the story is well understood, then the reader can answer questions concerning the events and relationships within the story. However, the measurement these researchers use is qualitative, rather than quantitative.

With story understanding programs such as BORIS (Lehnert et al., 1983), the researchers pose questions to the system and subjectively evaluate the answers to determine text comprehension effectiveness. But such evaluation is potentially biased because it is tempting to ask only those questions already known to be answerable. In contrast to externally posed questions, Chi (1995, Chi et al., 1989) reports that improved learning is correlated with human subjects who generate their own questions and explain the answers themselves (see also Pressley & Forrest-Pressley, 1985). This is the so called *self-explanation effect*. Thus, the ability of a system to pose self-generated questions both indexes actual understanding and simultaneously reduces the probability of asking only the easy questions.

Consider the following quote from Gavelek & Raphael (1985).

*One form of metacognition - metacomprehension - addresses the abilities of individuals to adjust their cognitive activity in order to promote more effective comprehension. We have been interested in a specific aspect of metacomprehension - namely, the manner in which questions generated by sources external to the learner (i.e., from the teacher or text), as well as*

---

113. In 1991, the Third Message Understanding Conference (MUC-3) cast the evaluation criterium for natural language systems as information extraction. The test is to provide a system with a uniform corpus of stories and to judge it based on how much information is extracted from it, how much of the extracted information is correct (as judged by prior human evaluation on the corpus), and how much extraneous information was extracted (Lehnert & Sundheim, 1991; Lehnert, Cardie, Fisher, McCarthy, Riloff, & Soderland, 1995). Although representing an unbiased approach, the criterium is more appropriate for evaluating text understanding (the performance system), rather than the learning, explanation, and metacognitive components of the performance system.

*those questions generated by the learners themselves, serve to promote their comprehension of text. (p. 104)*

The ability to adjust cognition in order to improve comprehension is at the heart of the research presented here. Thus, simply being able to recognize that a gap exists in one's own knowledge, and to therefore ask the question "Why don't I understand this?" (Ram, 1991), is the first step to improving the understanding, rather than actually giving an answer. So, to evaluate the ability of the performance of the Meta-AQUA system, credit should be given for simply posing a question that deserves asking.

A basic evaluation measure is to count the number of questions answered correctly to ascertain an absolute measure of performance. However, human students who are asked questions on reading tests are sometimes given points for partial answers. As discussed in the introduction, unlike questions in mathematics that have a provably correct answer, answers to explanatory questions are difficult to judge in an absolute sense. To make a more realistic measure, the criterium to evaluate performance in comprehension systems such as Meta-AQUA should assign some credit for providing any answer to a question.

Therefore, the full evaluation metric is as follows. For each anomalous or interesting input in a story, a point is given for posing a question, an additional point is given for providing any answer whatsoever, and a third point is assigned for answering what the researcher judges correct. The sum of these three point values represents the dependent variable.

### 9.2.3 The Empirical Data

This section reports data from six experimental runs of Meta-AQUA. To serve as experimental trials and to minimize order effects, Tale-Spin generated six random sequences of Elvis World stories (see Section 8.3.2, "The Elvis World," starting on page 185). On each of these runs, Meta-AQUA processed a sequence three times, once for each experimental manipulation. The system began all runs with the same initial conditions. For a given experimental condition, it processed all of the stories in the sequence while maintaining the learned knowledge between stories. At the end of the sequence, the system reset the BK. The input size for a run varies in length, but averages 27.67 stories per run.<sup>114</sup> The corpus for the six runs includes 166 stories, comprising a total of 4,884 sentences. The stories vary in size depending on the actions of the story and Tale-Spin's randomness parameters (e.g., the probability that characters throwing an object will stop on the current toss), but average 29.42 sentences apiece.<sup>115</sup>

### 9.2.3.1 Run Number Four

Run four is particularly interesting because the greatest number of negative learning interactions occur in this set. The input to run four consists of the 24 stories as enumerated in Table 9 (see Appendix C for a complete listing of all stories in this run). The stories contain a total of 715 sentences altogether. The average number of sentences per story is 29.8.

Each numeric entry in Table 9 contains a value for each condition. The entries are triples of the form <LG, RL, NL>. For example, the second column represents the number of learning episodes for each trial and for each condition. Note that the third element of each triple in this column is zero since learning is disabled in the NL condition. The sixth column (“Question Points”) contains the values for the dependent variable. These values represent the sums of the triples from the third, fourth, and fifth columns (“Posed Questions,” “Answered Questions,” and “Correct Answers,” respectively).

In this run, random drug busts occurs 11 times (5 with the canine squad and 6 with a lone police officer).<sup>116</sup> Table 10 shows the distribution of protagonists within the stories of run four.<sup>117</sup> Dad was the most common protagonist, while Elvis, Officer1, and Lynn were tied for the least common.

Table 11 shows the distribution of problems encountered by these main characters. Boredom is the main problem encountered in Elvis World for run four, although considering the number of random drug busts, the household can hardly be classified as sedate. The main characters solved (or attempted to solve) seven of these boredom problems by playing with one of three balls and solved three by playing with balloons. The state of being con-

---

114. The reason that each run varies in length is that, after generating around 600,000 gensyms, Meta-AQUA will use all available swap space on the Symbolics and thus inadvertently halt the underlying LISP system. We then discard the story which is being processed at the time of the crash. The data from the remaining stories constitute the results of the run.

115. As explained in Section 8.3, the input is preparsed and thus represented conceptually rather than in English.

116. Not every time the police dog accompanies the officer will he bark at inanimate objects. It barks at the location it thinks the pot is hidden. By random juxtapositions of events, this location will sometimes be animate. For example, story number three of run number four (see Appendix C for listing) contains a drug bust by the officer and his dog. The characters randomly enter the house just as Elvis is preparing to smoke by filling his pipe with marijuana. As a result, the police dog approaches Elvis and barks at him rather than at an inanimate object. The program thus detects no anomaly in the input dog-bark concept.

117. Karen was removed from the cast of main characters available as a protagonist, and the state of hunger was removed from the possible initial problem states, so that Tale-Spin now generates a more uniform distribution of failures.

Table 9: Results from run number four

Story Number (sentences) <sup>a</sup>	Learning Episodes (LG RL NL)	Questions Posed (LG RL NL)	Answered Questions (LG RL NL)	Correct Answers (LG RL NL)	Question Points (LG RL NL)	Protagonist and Problem <sup>b</sup>
1 (26)	1 1 0	1 1 1	0 0 0	0 0 0	1 1 1	Mom bored (balloon)
2 (19)	2 3 0	3 3 3	3 2 2	1 0 0	7 5 5	Mom bored (ball)
3 (38B)	1 1 0	1 1 1	0 0 0	0 0 0	1 1 1	Elvis jonesing
4 (51b)	0 1 0	1 1 1	1 0 0	1 0 0	3 1 1	Dad jonesing
5 (21)	0 1 0	1 1 1	1 0 0	1 0 0	3 1 1	Mom bored (ball)
6 (13)	0 1 0	1 1 1	1 0 0	1 0 0	3 1 1	Officer1 concerned
7 (13)	0 1 0	1 1 1	1 0 0	1 0 0	3 1 1	Dad bored (ball)
8 (21)	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	Dad thirsty
9 (44B)	1 2 0	2 2 2	2 1 1	1 0 0	5 3 3	Dad thirsty
10 (51B)	0 1 0	3 3 3	2 1 1	2 1 0	7 5 4	Dad bored (balloon)
11 (11)	1 2 0	2 2 1	1 1 1	1 0 0	4 3 2	Lynn bored (ball)
12 (3)	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	Officer1 concerned
13 (47b)	0 0 0	2 2 1	1 1 0	1 1 0	4 4 1	Mom thirsty
14 (15)	0 4 0	4 4 4	4 2 3	4 0 0	12 6 7	Mom bored (ball)
15 (28)	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	Lynn jonesing
16 (42B)	0 1 0	2 2 2	2 1 1	2 1 0	6 4 3	Dad jonesing
17 (45b)	0 0 0	2 2 1	1 1 0	1 1 0	4 4 1	Elvis jonesing
18 (21)	0 1 0	2 2 2	2 1 1	2 1 0	6 4 3	Officer1 concerned
19 (20)	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	Dad jonesing
20 (52b)	0 1 0	2 2 1	1 0 0	1 0 0	4 2 1	Dad bored (balloon)
21 (39b)	1 1 0	2 2 1	1 1 1	1 1 1	4 4 3	Lynn jonesing
22 (17)	0 2 0	2 2 2	2 1 1	2 0 0	6 3 3	Dad bored (ball)
23 (40B)	1 1 0	2 2 2	1 1 1	1 1 0	4 4 3	Elvis thirsty
24 (38b)	0 1 0	2 2 1	1 1 0	1 0 0	4 3 1	Mom bored (ball)
Total 715	8 26 0	38 38 32	28 15 13	25 7 1	91 60 46	

a. The letter “B” means that the story contains an attempted drug bust by the police canine squad, whereas the letter “b” means that the officer entered the house alone to attempt a bust.

b. Items in parentheses represent games played to dispel boredom.

Table 10: Main character distribution (run four)

Protagonist	Incidence
Dad	9
Mom	6
Elvis	3
Lynn	3
Officer1	3

Table 11: Problem distribution (run four)

Problem	Incidence
Boredom	10
Jones	7
Thirst	4
Concerned	3

cerned is the least recurrent problem exhibited in the run. This low frequency of occurrence is attributable to the constraint that Officer1 is the only character to whom Meta-AQUA can assign this state.

### 9.2.3.2 Quantitative Results

This section summarizes the results of the experiment. First, we will analyze the results of run number four, and then report a summary of all six runs. Table 9 totals the results in the bottom row. Table 12 presents these results. The Question Points column contains the value of the dependent variable. As shown in this column, Meta-AQUA performance under the LG condition is significantly greater than the performance under the RL condition. In turn, Meta-AQUA's performance under the RL condition far exceeds its performance under the NL condition.

Alternatively, if only absolute performance is considered as measured by the number of correct answers, then the differential is even greater, as shown in the fifth column. By this measure, the LG performance is more than three times the performance under the RL condition, whereas, the performance under the NL condition is insignificant. By looking

Table 12: Summary of results from run four

Learning Condition	Learning Episodes	Questions Posed	Answered Questions	Correct Answers	Question Points
LG	8	38	28	25	91
RL	26	38	15	7	60
NL	0	32	13	1	46

at column four, however, the numbers of questions answered in some way (right or wrong), are roughly equivalent in the RL and NL conditions, whereas the ratio of the LG condition to either of the other two is 2:1. Finally, the number of questions posed are approximately equal across all three conditions.

In contrast to these differences in performance, Meta-AQUA attempts to learn from failure more than three times as often under the RL condition as under the LG condition. That is, learning is more *effective* with learning goals than without. In the RL condition, learning does not increase performance as much as does the LG condition, while concurrently, it leads Meta-AQUA to expend more resources by increasing the amount of learning episodes. Thus, the system works harder and gains less under RL than under LG. After a closer examination of the trend in the dependent variable for run number four, this section statistically quantifies the increase in learning effectiveness across the six runs.

Figure 88 shows the accumulation of question points across trials (i.e., stories) in run number four.<sup>118</sup> This figure illustrates a clear trend that is representative of all six runs; the behavior of the system as measured by the dependent variable is greatest under the LG condition, next best under RL, and worst under the NL condition. However, the trend does not hold for each individual trial. For example, Figure 89, the raw scores, shows that the NL condition actually outperforms the RL condition in trial number 14. The reason for this effect is that under worse-case conditions, if the interactions present between learning methods are negative, the performance may actually degrade. As a result, randomly ordered learning may be worse than no learning at all.

The differences as a function of the independent variable are even more pronounced if only accuracy (the number of correct answers) is examined and partial credit ignored.

---

118. Note that the final extent of all three curves reach the value of the triple in the totals column for column six.

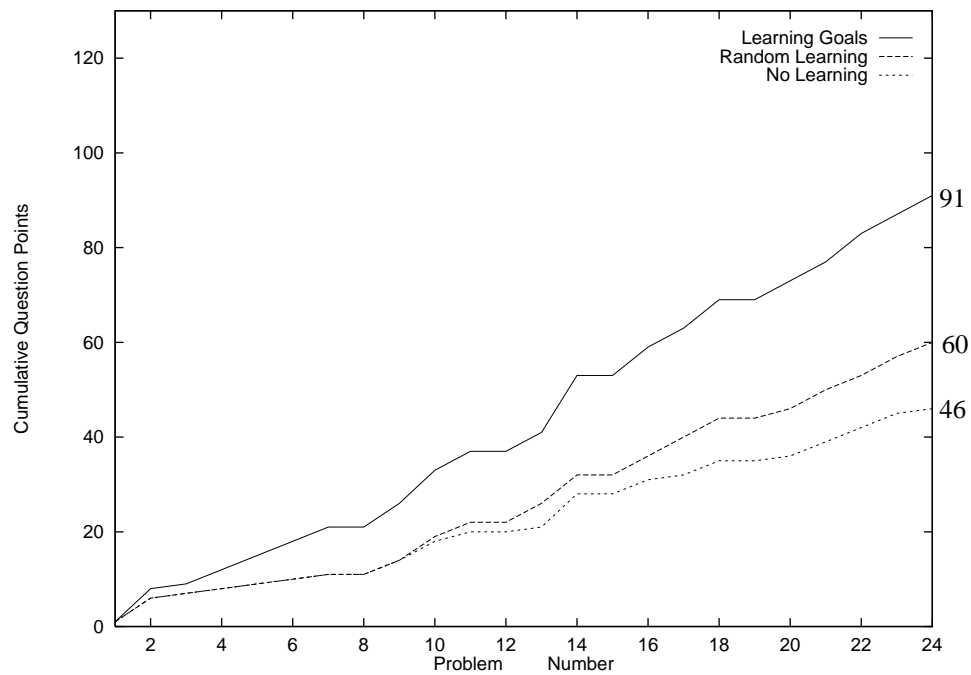


Figure 88. Run 4, cumulative question points as a function of the number of problems

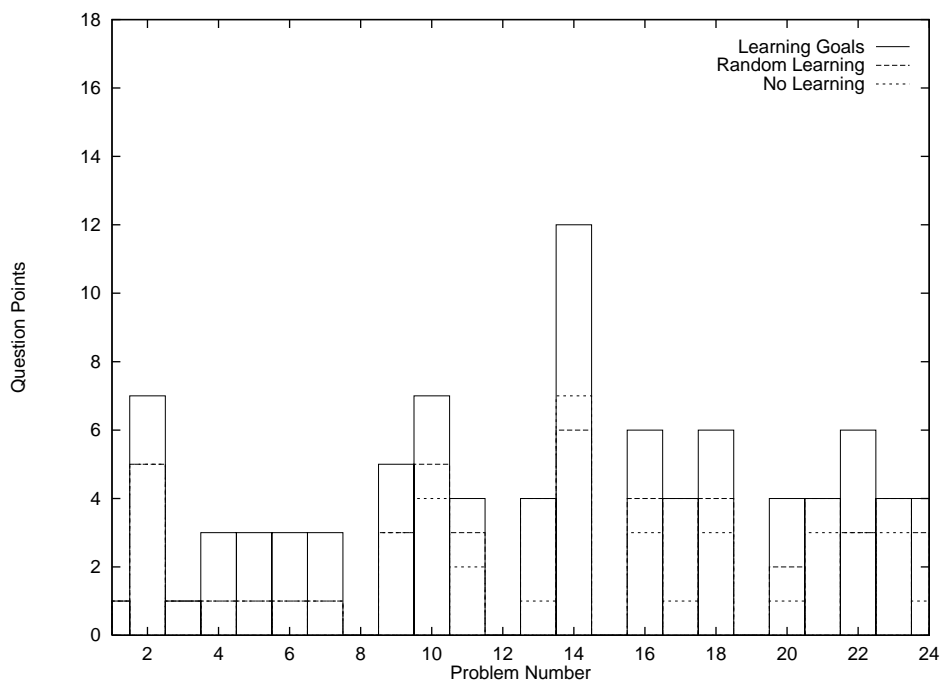


Figure 89. Run 4, question points histogram



Figure 90 shows that under the RL condition, Meta-AQUA did not answer a question correctly until trial number 10, whereas under the NL condition, it did not perform correctly until trial 21. On the other hand, because under the LG condition the system learned a new explanation early in trial number 1, it was able to answer a question by trial number two. This striking result was facilitated by the random order of input (i.e., the second trial happened to be about the same problem as the first) as well as by computational introspection. Figure 91 shows the raw absolute accuracy scores.

Table 13 summarizes the evaluation data from the six program runs. As is evident across all runs, the LG condition consistently outperforms the RL condition in the total cumulative question points. In turn, the RL condition outperforms the NL condition, despite the occasional poor performance due to negative interactions. As indicated by the standard deviations, the amount of differences between and within conditions exhibit high variability across runs.

Given these totals, the percent improvement for either learning condition over the NL base condition is simply the ratio of the difference in the base performance score and either score to the base score itself. Thus for run one, the ratio of the difference between the LG and NL conditions (35 points) to the NL condition (50 points) is .7, or 70 percent. Again, the improvement in performance for the LG condition is consistently higher than that of the RL condition. This difference is calculated in the final column. The differential is the percent improvement of the LG condition over the RL condition and is computed by the same measure as was the improvements in the individual learning conditions. That is, the differential is the ratio of the difference between the two improvements to the lower rate.<sup>119</sup> Thus, the differential between the LG rate of learning in run number one and that of the RL condition is the ratio of the difference (8 percentage points) to the RL percentage (62). Hence, the ratio is .129, or an improvement of nearly 13 percent.

Although the average differential between the two learning conditions (i.e., between fully introspective multistrategy learning and semi-introspective multistrategy learning) is more than 106 percent, this figure overstates the difference. The expected gain in learning is more conservative. The differential between the average LG improvement (102.70) and the average RL improvement (65.67) is a 56.38 percent difference. That is, across a number of input conditions, the use of learning goals to order and combine learning choices should show about 1.5 times the improvement in performance than will a straight mapping of faults to repairs when interactions are present.

---

119. Note that this ratio can also be calculated as the difference between the performance scores of the learning conditions to the difference between the performance score of the RL and NL conditions. In other words, the ratio  $(LG-RL) / (RL-NL)$ .

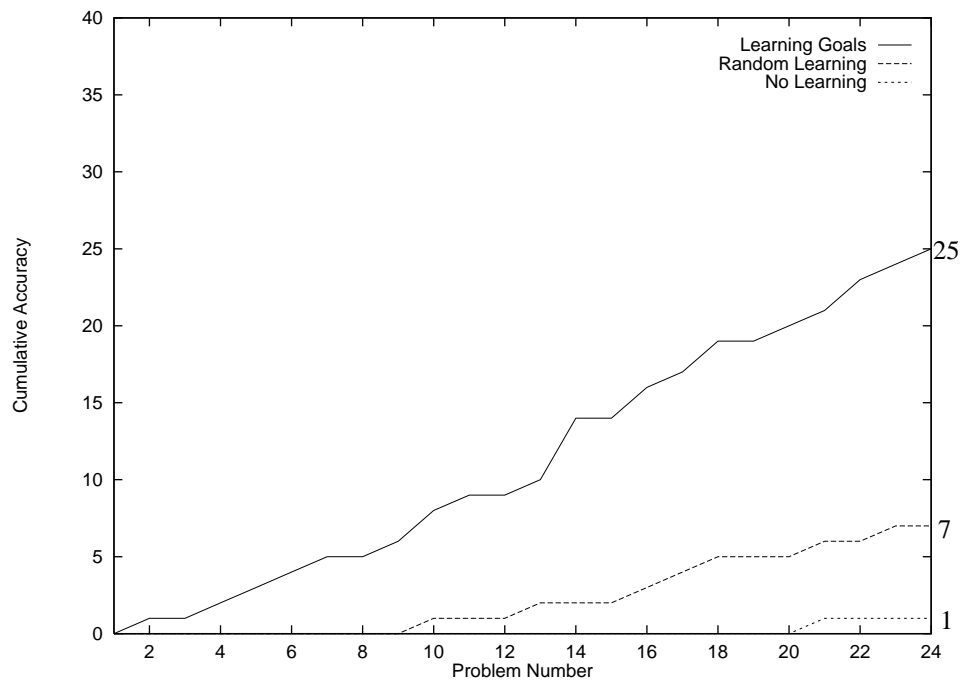


Figure 90. Run 4, cumulative correct answers (accuracy) as a function of the number of problems

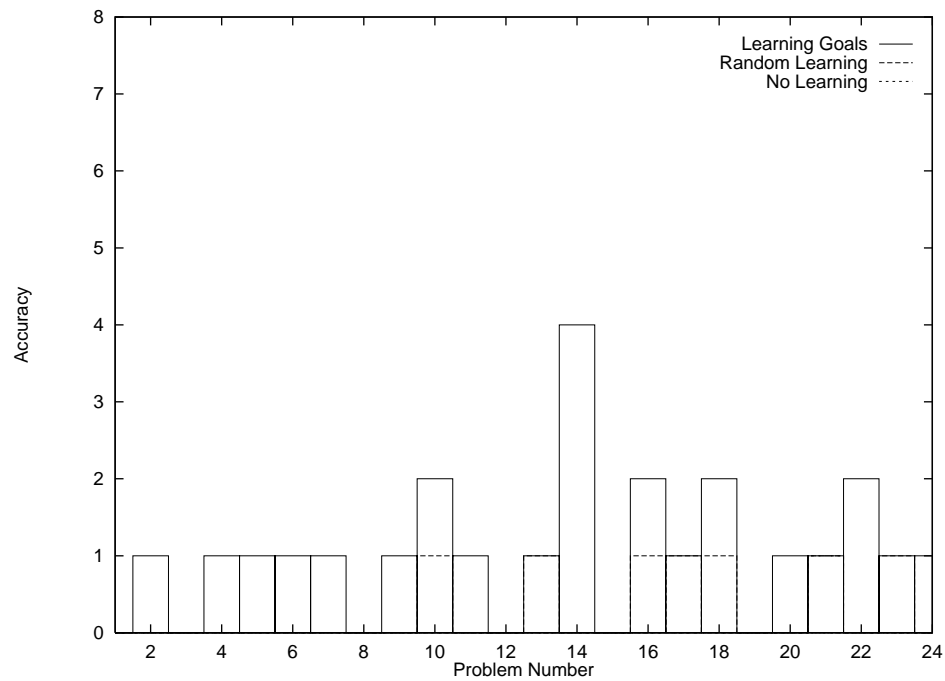


Figure 91. Run 4, correct answers (accuracy) histogram

Table 13: Summary of cumulative results

Run Number <sup>a</sup>	Cumulative Question Points			% LG Improved	% RL Improved	Improvement Differential %
	LG	RL	NL			
Run 1 (34)	85	81	50	70.00	62.00	12.90
Run 2 (30)	106	98	43	146.51	127.91	14.55
Run 3 (28)	120	102	60	100.00	70.00	42.86
Run 4 (24)	91	60	46	97.83	30.43	221.43
Run 5 (22)	57	49	27	111.11	81.48	36.36
Run 6 (28)	103	66	54	90.74	22.22	308.33
Averages	93.67	76.00	46.67	102.70	65.67	106.07
Std. Dev.	21.72	21.31	11.34	25.43	38.17	126.59

a. Amounts in parentheses indicate total number of stories in each run.

The results of this section show that learning is more effective with introspection than without, given the task of story understanding. Moreover, we have shown that because learning algorithms negatively interact, the arbitrary ordering of learning methods (i.e., as under the RL condition) can lead to worse system performance than no learning at all. Thus, an explicit phase to decide what to learn (i.e., via learning goals or an equivalent mechanism) is *necessary* to avoid such interactions in multistrategy environments. Without mediation between blame assignment and strategy construction, (i.e., without loose coupling) learning will not be effective given non-independent learning methods. The following section demonstrates that the content and process theories of introspective multistrategy learning presented in Parts Two and Three are sufficient to model introspection computationally.

### 9.3 Psychological Plausibility

The previous section tested hypothesis number one, examining Meta-AQUA from a computational perspective. This section tests hypothesis number two by examining Meta-AQUA to see whether it is sufficient for modeling human learners. Using protocols from the Pirolli and Recker study and taking the strategy categories as a given, the goal of this experiment is to begin to investigate the feasibility of modeling metacognitive behavior using IML theory. Although the ambitions of this section are more limited, for Meta-AQUA to fully model human metacognition it should be able to simulate the following imaginary sequence of reasoning. A graduate student fails his AI qualifying exam and must

retake it in another six months:

I flunked the depth exam because I did not know good answers for the cognitive science questions during the essay section and I forgot how the EBG algorithm and Michalski's Star algorithm really worked during the last part of the technical section (blame assignment). Thus, I need to learn more about both human psychology and machine learning algorithms beyond what was in the Ph.D. reading list (deciding what to learn). One way I can learn this is to find additional reading materials about some of the technical algorithms and I can take the graduate cognitive psychology course that I skipped last year. I can do a computer literature search to find papers dealing with machine learning in order to review the material I had in the machine learning class last quarter. I can then ask the other students that passed the test if they would suggest which papers are best, go to the library to either check them out or photocopy them, and finally read them thoroughly. I should then implement some of the algorithms so I remember them better. I could even run an experiment during the lab section of the Psych class. It could investigate how humans learn from examples and compare these results with the AI algorithms (learning strategy construction). I've got to pass the test next time.

As an experiment toward the goal of creating a system that reasons in similar ways to the graduate student, this section presents the results of an effort to simulate protocols of novices that are learning to program in LISP. This result is significant, not only because it is real instead of artificial data, but because the performance task is problem solving rather than story understanding. This bolsters the support for hypothesis two by showing that the theory is general enough to apply to more than a single task domain. After discussing the Pirolli and Recker data (Section 9.3.1) and the LISP programming simulation in Meta-AQUA (Section 9.3.2), a concluding section (Section 9.3.3) briefly discusses a separate IML-based implementation, called Meta-TS, that improves its ability to troubleshoot electronic circuits. This research adds support to the claim of generality of the theory.

### 9.3.1 Learning to program in LISP

Recker and Pirolli (1995) present a cognitive model based upon data from a study in which learners study instructional materials pertaining to LISP prior to problem solving in the programming domain. The instruction is embedded in a computer-based hypertext system, through which students navigate by clicking on mouse buttons. Learners attempt to understand the instructional material by explaining it to themselves, using a general learning strategy called self-explanation. After instruction, the subjects engage in problem solving in a recursive, LISP programming task using the CMU LISP tutor (Anderson & Reiser,

1985). The tutor provides intervention when the subjects make mistakes and eventually guides the students through each of the test exercises. It is important to note, then, that a given subject could actually finish a problem, thus producing a solution, without actually understanding the reasons why the solution is correct or sufficient for the problem.

Previous research has shown that a subject's engagement in self initiated explanation of examples appears to affect significantly a student's initial understanding and their subsequent problem-solving performance (Pirulli and Recker, 1994). Besides self-explanations, however, these studies report that introspective behavior also correlates with problem-solving performance. Between trials during the LISP programming tasks, subjects often generated reflective protocols spontaneously and without experimenter instruction or prompts. For example, subjects sometimes used awareness of comprehension failures as a cue to re-read example solutions or instructional text material. These kinds of learning strategies helped the subjects improve their subsequent performance by learning items such as new problem-solving operators.

Recker and Pirulli analyzed the subject protocols, thereby producing a categorization of the kinds of protocols subjects produce. In the most broad division, statements were of three types: Monitor, Domain and Strategy. *Monitoring statements* verbalize the subject's awareness of the level of performance or understanding. They typically assert whether they have a successful understanding, a failed understanding, or are unsure of their knowledge. *Domain statements*, on the other hand, concern the problem solutions and attempt to integrate them with their general understanding of the domain as acquired from the instructional text. Most protocols in this category either elaborate single problem solving episodes, or compare and contrast a recent one with either an earlier test problem, or an example problem from the instruction. The third protocol category, the Strategy type, is the major focus of this thesis.

*Strategy statements* are explicit statements concerning student intentions to learn. The seven subdivisions of this category (see Figure 92) relate how the students picture their performance with respect to their knowledge of the task and the instructions from which this knowledge was gleaned. An important goal in learning in the domain of programming is to make operational the declarative information in instructions, thus transforming propositional knowledge into specific procedures for writing code during the programming tests. The strategies in Figure 92 center on the above goal.

### 9.3.2 Meta-AQUA Simulation

This section describes the changes to Meta-AQUA required to model a selected protocol fragment from the Pirulli and Recker data. This exercise not only supports the sufficiency criterion for the cognitive plausibility of Meta-AQUA, but it also explores the

*1. Trace Restatement:*

This strategy is a superficial restatement of the solution in LISP-like terms. We have interpreted it as an attempt to rehearse the material, thus providing a better memory for the information therein.

*2. Re-read Solution:*

Re-reading the solution is a deeper review of the principle behind the solution, trying to understand it in an abstract sense.

*3. Reflect on Errors Made:*

This reflective strategy attempts to learn by understanding what went wrong with a problem-solving move that the tutor corrected.

*4. Write Down Solution:*

Committing solutions to paper is taken as an attempt to use an external representation as an extension of working memory and as an attempt to rehearse and review the material.

*5. Re-read Text:*

Often because additional information is obtained during the reading of an example that is useful in better understanding the instruction, a rereading of the text after this information has been acquired can provide a better understanding of the information.

*6. Hypothesize Alternative Solution:*

Because there are multiple solutions to any given problem, often subjects find it useful to pose alternate solutions to a recent problem. By elaboration of the ways in which success can be obtained, a more general understanding can be derived.

*7. Identify Comprehension Failure:*

Subjects sometimes know that they fail because of the tutor feedback, but they need to specifically search for the knowledge that was incomplete or incorrect.

---

Figure 92. Strategy protocol categories

usefulness of the model in understanding the role of reflection in learning to program.

The focus of Meta-AQUA, however, is neither on the understanding tasks nor on the problem-solving tasks involved in programming. Rather, the central questions explored pertain to events that occur when problem-solving or understanding fail. As a result, Meta-AQUA places an emphasis upon examining a trace of the reasoning processes that preceded the failure in order to determine the causes of failure. Thus, the central concern of the model is reflective processing, similar to that which may underlie the kinds of reflection exhibited by the subjects in the Pirolli and Recker study.

Meta-AQUA's principal domain centers on understanding stories of drug smuggling using knowledge given from the original AQUA implementation in the domain of terrorism. However, in addition to the story-understanding mode, it also has an alternate problem-solving mode that models two agents, a smuggler and a drug-enforcement agent, interacting in a cat and mouse scenario. This mode was initially designed to test the potential of integrating understanding and problem-solving. When verifying a hypothesis in an understanding task an agent might actually devise a test through problem solving, and when trying to verify a plan in a problem-solving task an agent might verify it by running the plan to understand whether the plan met his expectation during execution in the world. This mode was modified to implement the LISP programming task in the Pirolli and Recker study.

We model the subjects as problem-solvers who plan to achieve programming goals. They then attempt to run the program and check whether it meets their expectations during execution time. IML theory's central notion of expectation failure provides a basis for modeling comprehension failures exhibited by the subjects. The changes to Meta-AQUA necessary to simulate a protocol in LISP programming study were slight. The major change was an addition of simple simulator for the LISP Tutor that provided feedback to the programmers. All other additions consisted of building the knowledge-representation frames for LISP functions, programming plans, and other objects in the programming world that the system processed (see Appendix D, "META-AQUA OUTPUT IN LISP PROGRAMMING MODE."). No IMXPs were added to the system for the modification.

Because Meta-AQUA is concerned with the interaction of multiple learning choices, we chose a protocol that combines three reflective strategy protocols. Subject AK88's protocol was taken from data reported in Pirolli & Recker (1994). The subject was given the task of creating a recursive LISP function, called `Add1num`, that would take as input a list of integers and letters, outputting as a result a list of only the numbers incremented by one. For instance, given the list '(1 3 a 4 b 1), the function should return the list '(2 4 5 2). Figure 93 details a fragment of the protocol from subject AK88 after performing this problem, but before the computer system loaded and presented the next problem.<sup>120</sup>

[18:28 After solving ADD1NUM problem]

Oh I see, I would have to use 1+, I could use Add1nums because for this, for every thing but the first element. Unfortunately, it's not clear to me how this works... I'm not sure why you would have to, you would have to code in the beginning um...1+ you have to add 1 to the first element of the list but for all the other elements I could just use add1nums.

(## strategy # Id-comp-failure)  
(## strategy # reflect-on-err)

I'm going to try to look through lesson five because I remember there's a section that says that you can use an example.

(## strategy # reread-txt)

[rereads "Useful Procedures" page of instructions]

---

Figure 93. Protocol fragment of subject AK88

We interpret this protocol as being composed of three learning strategies. First, AK88 becomes aware that her comprehension of recursion is not correct. This is a strategy we call **Identify Comprehension Failure**. In Meta-AQUA this sequence is represented by the structure in Figure 94. The subject does not understand that the correct LISP function should add one (using the 1+ increment function) to the first item in the list if it is a number and then **CONS** that to a recursive call of the function on the rest of the list. Instead, the student appears to be confused about the recursive part of the function, preferring to use the increment function (1+) repeatedly. See Figure 95 for the correct code that should be generated and the possible construction that the subject is attempting to use.

Secondly, to understand this fact AK88 tries to reflect over the error made after realizing that there was indeed a comprehension failure (see node labeled **Reflect on Error** in Figure 94). Thirdly, the subject is reminded of a prior example in the instructional text and decides to review it by rereading the text (node labeled **Reread Text** in Figure 94). These fragments are particularly interesting, not only because they are consistent with much of the Meta-AQUA implementation and theory, but because they show a composite strategy that combines three separate components. Currently, Meta-AQUA was only able to simulate the construction of the top one third of the structure shown in Figure 94. Appendix D

---

120. Bracketed statements are contextual information, whereas parenthetical text specifies the strategy type as judged by multiple, blind raters.



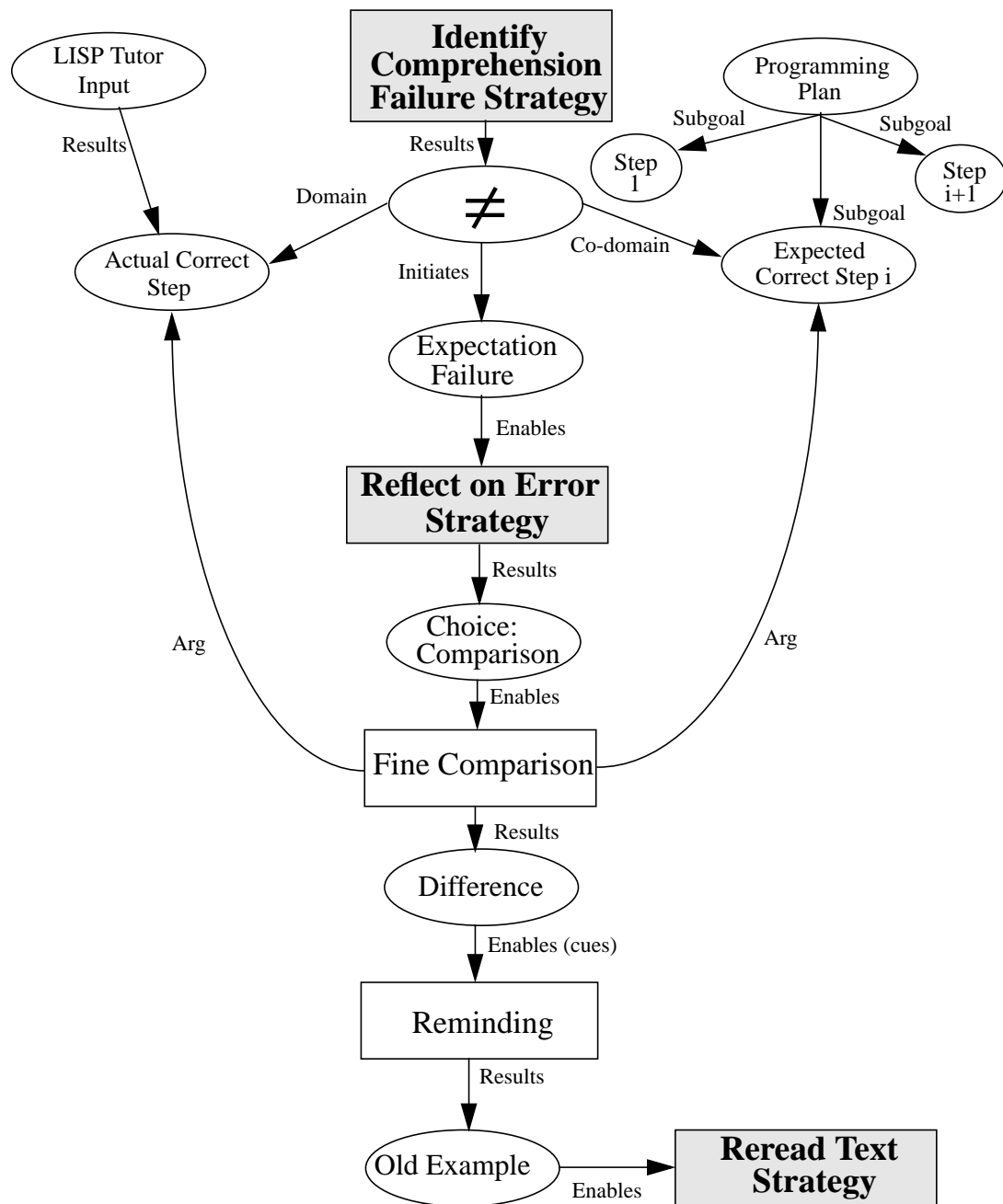


Figure 94. Representation for interaction of learning strategies of protocol fragment from subject AK88.

**Correct Recursive Approach:**

```
(defun Add1nums (alist)
  (cond ( (null alist) nil)
        ( (numberp (first alist))
          (cons (1+ (first alist))
                (Add1nums (rest alist))))
        ( t
          (Add1nums (rest alist)]
```

**Flawed Iterative Approach:**

```
(defun Add1nums (alist)
  (cond ( (null alist) nil)
        ( t
          (list
            (if (numberp (first alist))
                (1+ (first alist))
                nil {nothing})
            (if (numberp (second alist))
                (1+ (second alist))
                nil {nothing})
            ....
            (if (numberp (last alist))
                (1+ (last alist))
                nil {nothing}))))]
```

---

Figure 95. Add1nums function definitions

shows output from the program and the goal structure that is pursued during execution of the simulation. Figure 96 shows three of the frame definitions used to represent the flawed iterative function presumed to be considered by subject AK88.

Although future research is necessary to firmly validate this approach, the results look promising. Meta-AQUA modeled most of the protocol sufficiently, both with the subjects cognitive task (problem-solving) and the subject's meta-cognitive task (strategy construction). However, because the problem-solving mode of Meta-AQUA is not fully implemented the simulation of the programming task was overly simplified. And because the system's memory module is incomplete, the section of the protocol pertaining to the subject's reminders of a previous problem solving episode (i.e., a programming problem) was not finished. Therefore, to finish this trial with Meta-AQUA, the system must be programmed with both a more robust problem-solving process and a more plausible memory. Although the above implementation is preliminary, the next section adds to this evidence in support of the sufficiency claim.

### 9.3.3 Learning to Troubleshoot Circuit Boards

Based on IML theory, we have implemented an independent system that simulates human protocol data involving novices that learn to troubleshoot faulty circuitry by modi-

```

;;; NOTE that the beginning of the definition is just like the definition of a MOP (Memory Organization Packet).
;;; There is one important difference between MOPs and LISP-functions, however. The object that is manipulated
;;; is the precondition state, instead of being something separate. One could think of the preconditions as being some
;;; kind of test on the objects that are passed to the function, but this analogy holds only partially. Thus depending on the
;;; values of the various roles in the structure one could have different variation of a function.
;;;
(define-frame LISP-FUNCTION
  (isa (value (computer-command LISP-program)))
  (actor (value (programmer)))           ;; AK88
  (preconditions (value ((state))))      ;; A list of one or more states.
  ;; Note that, depending on branching, this may be
  ;; determined only at run-time, not statically like MOPs.
  (goal-scene                             ;; But usually not known until run-time.
    (value (LISP-function
      (main-result
        (value =main-result))))))
  (scenes (value (=goal-scene)))          ;; And other scenes are of course possible.
  (main-result (value (state)))
  (side-effect (value (state)))
  (semantics (value (literal)))           ;; How the function will behave.
  (parameter-list (value =preconditions))
  (program-steps (value =scenes))
  (return-value (value =main-result)))

(define-frame DEFUN-CALL
  (isa (value (LISP-function)))           ;; Inherits from LISP-function definition above.
  (name (value (literal)))                ;; Function name
  (lambda-list (value ((entity))))        ;; Argument list
  (body (value ((LISP-function))))        ;; Function body
  (parameter-list (value =name =lambda-list =body))) ;; The guts of the LISP function

;;; The following definition represents AK88's attempt to do recursion. The student is using a kind of iterative approach:
;;; Make an output list with the incremented value of the first item of the input as the first element, the incremented value of
;;; the second item in the input as the second element, and so on.
;;;
(define-frame NEWLY-LEARNED-ADD1NUMS-DEFUN
  (isa (value (defun-call)))               ;; Inherits from defun-call definition above.
  (actor (value (programmer)))             ;; The agent performing the coding.
  (name (value (literal "Add1Nums")))      ;; Recursive-add1nums-defun wanna-be.
  (lambda-list (value ((entity))))         ;; Function arguments. Can be arbitrary in number at run-time.
  (body (value
    ((cond-function
      (scenes
        (value
          ((conditional-clause
            (test-clause
              (value (null-op (parameter-list (value =lambda-list))))
            (action-clauses (value (nil.0))))
          (conditional-clause
            (test-clause (value true.0))
            (action-clauses
              (value ((constructor-op
                (first-item (value
                  (increment-op
                    (parameter-list
                      (value (first-op
                        (parameter-list
                          (value =lambda-list))))))))
                (old-list (value
                  (list-op
                    (parameter-list
                      (value (increment-op
                        (parameter-list
                          (value (second-op
                            (parameter-list (value =lambda-list))))))))
                    ))))))))))))))))
    ))))
  ))))

```

Figure 96. Frame definitions to represent newly learned programming knowledge

ifying, acquiring and deleting associations and heuristics used to formulate shallow explanations for circuit board behavior. The system, called Meta-TS<sup>121</sup> (Narayanan, Ram, Cohen, Mitchell & Govindaraj, 1992; Ram, Narayanan, & Cox, 1995), models the learning extracted from protocols of actual novice troubleshooters at an electronics assembly plant in Atlanta, Georgia. The model is based on protocol analysis of over 300 problem-solving episodes gathered at the in-circuit test facility of an NCR plant located near Atlanta, GA (Cohen, 1990; Cohen, Mitchell & Govindaraj, 1992).

The problem-solving module of Meta-TS uses different types of knowledge and test procedures to hypothesize the cause of a failure and to suggest repair actions for that failure. Based on our data from the human operators in the NCR plant, Narayanan et al. (1992) categorized diagnostic knowledge into two broad types. Associations are simple rules which directly map a particular symptom to a specific diagnosis. This type of knowledge is context-sensitive and is indexed by board type. Heuristics are standard rules of thumb. These rules are not context-sensitive and are applicable across board types. Heuristics are used by the operator for troubleshooting when there is no known association for a given problem situation. This knowledge determines the series of standard operating procedures performed in troubleshooting a faulty circuit board. Since the problem-solving module relies on associative and heuristic knowledge, the learning module must, in general, be able to acquire, modify, or delete such associations and heuristics through experience.

The introspective learning module of Meta-TS has several strategies for learning associative knowledge for the troubleshooting task, including unsupervised knowledge compilation, supervised learning from an expert, postponement of learning goals, and forgetting invalid associations. The first strategy is that of unsupervised, incremental inductive learning, which creates an association when the problem-solving module arrives at a correct diagnosis using heuristic knowledge. The introspector compiles the heuristic knowledge into an association. The second learning strategy creates a new association through supervisory input. This strategy is triggered when the system arrives at an incorrect solution using heuristic and/or associative knowledge. A third learning strategy is that of postponement. This opportunistic strategy is triggered when the system is unable to get immediate input from a skilled troubleshooter. The system posts a learning goal, keeps track of the reasoning trace for the particular problem-solving episode, and asks questions at a later time to gather appropriate associative knowledge. Two deletion strategies remove associative knowledge when it is no longer valid. The first strategy uses expert input to delete associations, and is invoked at the end of every problem-solving episode. The second deletion strategy is unsupervised and is selected when Meta-TS arrives at an incorrect solution and the reasoning trace shows that a single association was used in arriving at the solution.

---

121. TS stands for troubleshooter.

Meta-TS was evaluated using 42 actual problem-solving episodes gathered at the electronics plant over a 2-month period. The problems dealt with various types of resistor failures and are representative of the types of problems encountered over the period. To evaluate the learning methods, we tested the following five conditions on the 42 test problems.

- **H (hand-coded):** The original non-learning system with hand-coded associations. This condition represents a troubleshooting system that has been hand-designed by an expert researcher, and is useful as a benchmark in determining the strengths and limitations of the learning strategies.
- **NL (no learning):** The system with all associations removed and learning turned off. This condition represents a base case against which to evaluate the efficacy of the learning strategies; it uses only heuristic knowledge.
- **L (learning):** The system with all associations removed and learning turned on. This is the basic Meta-TS system with no prior experience.
- **L42:** The system with all associations removed, then trained it on the 42 test problems with learning turned on. The system was then evaluated by re-running it on the same 42 problems. This condition was intended to validate the learning strategies in Meta-TS by ensuring that they learned the knowledge required to solve the problems.
- **L20:** The system with all associations removed, then trained on 20 randomly generated training problems with learning turned on. The system was then evaluated by running it on the 42 problems.

Each of these conditions were evaluated quantitatively for speed and accuracy on the 42 test problems, and also qualitatively by examining the content of the learned knowledge and details of the solution process. Figure 97 shows the cumulative accuracy of the system for the various conditions. The H condition arrived at the correct diagnosis in 86% of the 42 problems. The L42 condition arrived at the correct diagnosis in 81% of the problems. The values for the L20, L, and NL conditions were 76.8%, 76%, and 71% respectively. Figure 98 compares the speed of the solution process (measured by the number of intermediate steps) with the various learning conditions relative to the hand-coded condition. The L20 and L42 conditions consistently arrive at the diagnostic result faster than the H condition. The L condition takes about 20 problem episodes to reach the same speed as that of the H condition and then consistently arrives at the diagnostic result faster than the H condition. At the end of the 42 problem episodes, the ratios of the learning conditions to the hand-coded conditions are: 1.52 (L42 to H), 1.24 (L20 to H), and 1.06 (L to H).

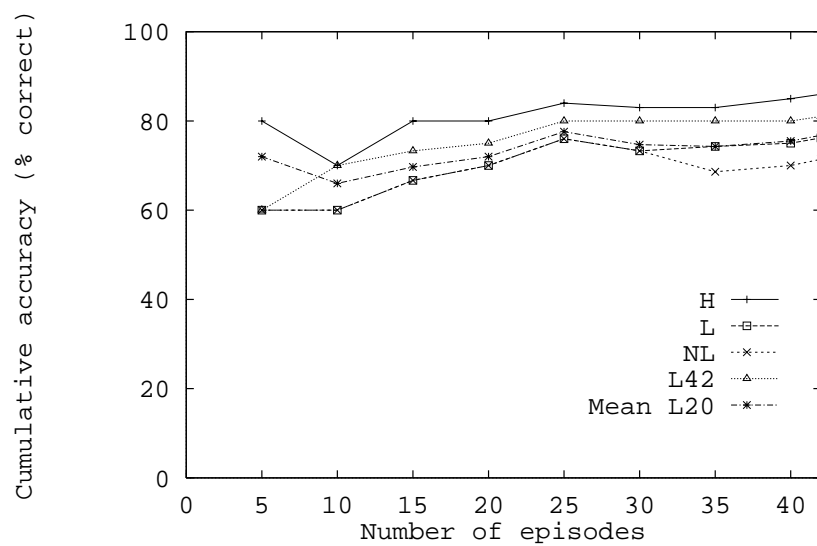


Figure 97. Cumulative diagnostic accuracy  
(From Ram, Narayanan & Cox, 1995)

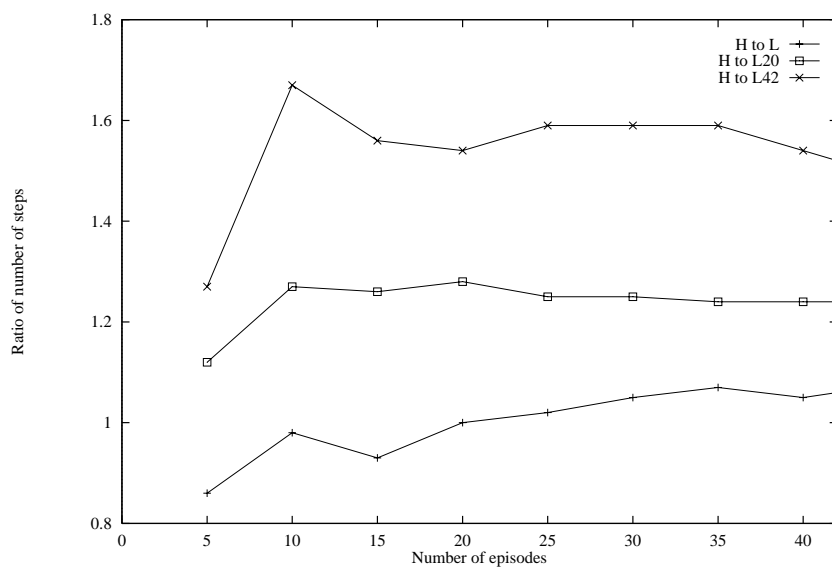


Figure 98. Speed ratio of learning conditions to hand-coded condition  
(From Ram, Narayanan & Cox, 1995)

The multistrategy learning module in Meta-TS clearly contributes to enhanced system performance in the troubleshooting task. In comparison with the non-learning system with no hand-coded associations, the associative knowledge learned by Meta-TS increases the accuracy of the diagnostic result and speeds up the problem-solving process. The performance of Meta-TS further increases when it is trained on similar problems before it is applied to novel problems. The associative knowledge learned by Meta-TS enables it to arrive at the same solution as that of the system with the hand-coded associative knowledge between 89% and 94% of the time.

Both the Meta-AQUA and the Meta-TS implementations have demonstrated to be sufficient models to approximate the learning of novices in a problem-solving situation. This validates the theory in a number of ways. First, the implementations have simulated real human data rather than simply appealing to the intuition of the reader. Secondly, the theory has been applied to three different task domains: learning to understand stories, learning to program a computer, and learning to troubleshoot electronic circuits. It therefore represents a general approach to learning rather than a specific one.<sup>122</sup> Third, the theory works using both deep causal explanations (as with the Meta-AQUA implementation) and shallow causal reasoning (as with Meta-TS). Moreover, results from the metacognition literature also support the idea that such a model is a reasonable one with which to understand deliberate learning. In particular, Brown (1987), Nelson & Narens (1994), and Weinert (1987) all make strong claims that metacognitive activities are intimately and positively interrelated with learning and understanding processes. Given these results, IML theory can be considered a reasonable model with which to understand metacognitive processes in human learning.

## 9.4 Summary and Discussion

The experiments discussed in this chapter provide a number of results that support the hypotheses established at the beginning of this dissertation. Evaluating Meta-AQUA with and without learning goals generated empirical results indicating that computational introspection (as defined in IML theory) facilitates the learning process. In particular, the results lead to the conclusion that the deciding to learn stage that posts learning goals is a necessary stage if negative interactions between learning methods are to be avoided and if learning is to remain effective. The chapter has also provided a novel quantitative measure with which to evaluate the comprehension process. As dependent variable, this partial credit metric provides rewards for both posing questions and giving some type of answer, as well as getting the answers right. Finally, the last section showed the generality of IML theory by

---

122. As will also be remembered from Chapter VII, the same IMXPs have been used to understand stories about both criminal activities as well as sports activities. See story HC1' on page 170.

reporting two additional tasks to which the theory applies (programming in LISP and electronics troubleshooting). The minimal modifications necessary to get Meta-AQUA to cover the LISP learning protocol fragment suggests that IML theory is a sufficient model of introspection. The results presented in this chapter also support the hypothesis that the failure taxonomy as described in Section 3.2 on page 45 is a reasonable categorization for both artificial and natural reasoners because these failure types are instrumental in the Meta-AQUA system from which the results were obtained. In particular, see the discussion in the summary section of Chapter III.

Research concerning introspection has long been controversial (e.g., see Boring, 1953; Nisbett & Wilson, 1977 for objections to such research). In the early days of psychology, trained introspection was assumed to be the proprietary scientific tool of the psychologist when “objectively” studying the mind.<sup>123</sup> The behaviorists tried to erase all scientific association with introspection by claiming not only that learning should be examined without the use of such introspective methods (e.g., Watson, 1919), but moreover that learning should be explained without reference to any intervening mental variables whatsoever (e.g., Skinner, 1950, 1956). Under the banner of metacognition research, however, interest has recently returned to the study of introspection, second-order knowledge, and their roles in cognitive activities (see Section 11.2 on page 272<sup>124</sup>). Yet, to believe that introspection is a kind of computational panacea is a deceptive assumption. Wilson and Schooler (1991) have empirically shown that conditions exist under which introspection actually degrades specific performance (e.g., preference judgements). In the context of story understanding, Glenberg, Wilkinson, & Epstein (1982/1992) reported that human self-monitoring of text comprehension is often illusory and overestimated, especially under the conditions of long expository text. Likewise, computational introspection is not effective under many circumstances, and, given the demonstrated limitations of human introspection, IML theory tries not to overstate its scope. Rather, as asserted at the very beginning the thesis, one of the goals of this research is to determine those conditions under which introspection is useful, and those under which it is not.

Because of the considerable computational overhead involved in maintaining a reasoning trace, performing blame-assignment, spawning learning goals, and constructing a

---

123. Titchener and others took great pains to develop a rigorous method of introspection and attempted to equate it with objective inspection (observation) as practiced in physics. For example, Titchener (1912) claims that “Experimental introspection, we have said, is a procedure that can be formulated; the introspecting psychologist can tell what he does and how he does it.” (p. 500). This remarkable statement is at the same time naïve and arrogant, given the hindsight of history.

124. But in the interim, see Lieberman’s (1979/1992) article entitled “Behaviorism and the mind: A (limited) call for a return to introspection.”



plan with which to pursue such goals, the benefits of using introspection must be substantial to justify the costs.<sup>125</sup> Furthermore, under extremely complex situations or in informationally impoverished circumstances, deciding on an optimal learning goal is certainly intractable. In such situations, it may be more beneficial to proceed without further reasoning, rather than to attempt to understand the exact causes of the failure. Knowing when a problem is worth pursuing is itself an important skill to master for an intelligent system. Identifying the most appropriate conditions for the use of an introspective approach is therefore a desirable research goal. To establish only that introspection facilitates learning and that the model of introspection has some quality of reasonableness is not satisfactory. A full evaluation will not only show the hypothesis to be valid under a particular circumstance, but will demonstrate the conditions under which these hypotheses hold. Although an experimental inquiry into these conditions will be left for future research, a number of remarks can be made at this time.

If the distributions of the kinds of failures generated by the performance task change the nature of the differences in the learning curves generated in the experiments used to establish hypothesis number one (see Section 9.2.3), then applicability conditions can be established that predict when the utility of introspection exceeds its cost. The space of applicability conditions for introspection is expected to emerge from the causal factors as organized by Table 5, “Detailed taxonomy of causes of reasoning failure,” on page 53. It has already been shown through the existing implementation that introspection in certain circumstances is tractable. Thus, a lower bound is already available. It is clearly not possible to reason in any effective manner if all possible failures occur at once. So, an analysis of the interaction of the causal types in the table should result in a set of complex failures that can be programmed into Tale-Spin in order to produce various distributions of errors. Meta-AQUA is expected to have difficulty learning from some of the failure combinations within these error distributions. As with the ablation study, measures with and without introspection provide the independent variable for the evaluation of learning. The results should itemize the conjunctions of failure from which it is impossible to recover and those for which a simpler reflexive or tightly coupled approach is more suited.

An additional reason that we expect such a test to work is that, as already demonstrated by this chapter and by Chapter VII, all learning methods are not independent. When they interact there will be substantial detriment to performance. Thus, one simple study would be to add a parameter to Tale-Spin that can limit the types of failures to those whose learning repairs are known not to interact. In this condition, the learning curves for the LG

---

125. One must be cautious, however, when dismissing introspection simply because of computational overhead costs. Doyle (1980) warns that to disregard the introspective component and self-knowledge in order to save the computational overhead in space, time, and notation is discarding the very information necessary to avoid combinatorial explosions in search (p. 30).

and RL conditions should not significantly differ. This will formally establish that at least one condition exists within which the utility of learning goals is lessened. However, to establish the quantitative trade-off between the utility of introspection and the cost, a more formal estimation for the cost must be constructed.

In the interim, a potential heuristic for deciding when to use an introspective approach is to qualitatively ascertain whether or not interactions between learning mechanisms available to the learner exist. If they exist, then the approach should be applied, otherwise a more reflexive approach is licensed. In speculation, another potential heuristic for determining that introspection is a win is to use a threshold for the number of failure symptoms above which introspection will not be attempted. Through experimentation, this threshold number should be obtained empirically given a distribution of known problem types and a random selection of problems from the distribution. The identification of such heuristics will enable the practical use of introspective methods in systems that cannot afford to squander precious resources with intractable computation.

## CHAPTER X

### FUTURE RESEARCH

*Man is not the sum of what he has but the totality of what he does not yet have, of what he might have.*

—Jean Paul Sartre (1939)

The effort to construct the theory and implementation presented in this document has been substantial and, as a result, a few questions have been left open to further inquiry. Along the way this document has pointed out both the deficiencies of the implementation and the open theoretical issues remaining for future research. This chapter attempts to gather these comments into a single space and to examine them with the intention of devising a plan for future resolution. Additionally, some related avenues of research exist that share many of the issues pertaining to IML theory. The following pages show how the foundation presented in the previous chapters can serve as a basis for the continuation and extension of the current research and can provide a foundation upon which to apply the research to learning problems involved in training and education.

The current chapter proposes a number of lines of investigation that extend the current research. Section 10.1 continues the effort of treating learning as a planning task and builds directly on the work presented here. The goal is to further formalize the computational task of learning-strategy construction, important in multistrategy learning systems that integrate multiple learning algorithms, and to further develop IML theory as a model of human learning. Section 10.2 extends IML theory by posing a novel approach to learning conceptual categories represented as Horn-clause propositional logic representations. This section also speculates as to what extent such work can transfer to hierarchical case-representations. We propose to develop learning algorithms that compare and contrast both expected and actual categories when revising background theories used in classification tasks in a manner similar to humans. Finally, Section 10.3 proposes an applied research project that uses the IML algorithm to understand student explanation failure in the context of intelligent tutoring systems or learning environments. The research that enables a machine to reason about its own explanation failures should easily be adapted to enable a machine to reason about user's explanations failure during learning. The research directions in all three

sections combine ideas from artificial intelligence and cognitive science and have the potential to make major contributions to both fields.

## 10.1 IML Theory and Implementation: Open issues

The research discussed in this section represents a direct interpolation of the current research. Table 14 lists the issues raised in previous sections necessary to make IML theory and its implementation in Meta-AQUA more complete. For each issue, the table lists the sections and page numbers on which the issue was discussed. This section examines each in turn, briefly describing the problem and proposing an approach for its solution.

Table 14: Extending IML theory

Issue for Future Research	Section(s)	Page(s)
Increasing the scope of blame assignment	VIII <sup>a</sup>	178
Additional goal interactions	7.4	170
New learning algorithms for toolbox	7.5, 8.5.1	173, 197
Parallelism in learning plans	7.4.2, 7.5	172, 173
Learning new IMXPs	6.2	136
Auto-validation of learning	5.4.2, 7.5	121, 173
Failure types as a cognitive category	3.4	65
Extend the computational evaluation	9.1.1.2, 9.4	217, 243
Extend the modeling of human learning	9.3.2	236
Extend the representational vocabulary	4.8	101

a. The discussion is contained in the introduction to Chapter VIII.

### 10.1.1 The Scope of Blame-Assignment.

As mentioned in Chapter VIII, the Meta-AQUA implementation considers only a limited number of causes for failure during blame assignment. This represents one of the most pressing needs for future research, that is, to expand the number of cells in Table 5 considered by the blame-assignment phase of learning. The amount of work required for this task will be significant because the institution of such change requires many extensions and additions.

For instance, new IMXPs must be developed to explain the failures and to attribute blame. In order to accomplish this task additional representation vocabulary must be established to declaratively represent the IMXPs (see Section 10.1.10). Furthermore, new learning-goal types must be created that can pinpoint the potential causes. In response, an analysis of the interaction of these goals must be performed (see Section 10.1.2). If new learning goals are included, then additional learning algorithms must be added to the toolbox to solve these learning goals (see Section 10.1.3). Therefore, to adequately expand the blame-assignment task, a number of the projected future research topics must be simultaneously considered.

### 10.1.2 Additional Goal Interactions.

One of the claims of this research is that learning is like planning. The planning metaphor has been successfully used in the Meta-AQUA system to detect learning interactions between learning steps within an overall learning strategy. Chapter VII has demonstrated that at least one classic planning-goal interaction (brother-clobbers-brother-goal, Sussman, 1975) does indeed fit a learning interpretation of the metaphor. But to more fully determine at what points the planning metaphor fits learning problems and at what points it does not fit, a survey of the planning literature (e.g., Allen, Hendler, & Tate, 1990) can examine other classic goal interactions to see if they too apply in a learning environment.

For example, Wilensky (1983) analyzed a number of positive and negative goal interactions that exist in conventional planners. Resources limitations (e.g., time or computational resources) and mutually exclusive states are two negative relationships whereas goal overlap is a positive goal relationship. It seems reasonable that all of these interactions will have a learning goal interpretation. As one more example of goal conflict, suppose that a particular algorithm evaluates a concept's expected utility, and if it is lower than some threshold, deletes it from memory. Another algorithm may generalize the same concept. If the generalization is run before the forgetting (deletion) algorithm, then the expected utility may be increased so that the item is not deleted. If the algorithms are executed in the reverse order, the generalization algorithm will find nothing left to generalize.

Finally, this line of inquiry also depends on more fully developing the taxonomy of learning goals. Not only are additional acquisition goals a possibility, but, as mentioned in Section 6.3.1 (see footnote 79), there are certainly circumstances where a learning operator will need to establish a prevention goal so that particular states of the BK are not acquired by the system or a maintenance goal established so that a particular belief or proposition is preserved.

### 10.1.3 Enlarging the Toolbox.

The number of algorithms presently contained in Meta-AQUA's toolbox is quite limited. Only four have been implemented at this time. They are case acquisition, EBG, abstraction, and index learning. To scale the system to larger and more difficult problems, and to better understand this approach to multistrategy learning, the size of the toolbox must be increased. In support of this goal we intend to expand the number of learning algorithms contained in Meta-AQUA's learning strategy toolbox. An early candidate for inclusion is the ID3 algorithm (Quinlan, 1986) that learns through decision trees.

At the current time, we have addressed learning-strategy construction in only a limited manner. That is, we have concentrated on integrating learning algorithms that do not perform the same learning function. Therefore, to fully address learning-strategy construction it is necessary to address the selective superiority problem (Brodley, 1993). This is a very difficult problem that involves choosing the best learning algorithm (as originally formulated, the best inductive algorithm) with which to solve a given problem with a specific distribution of input data. In some instances, it appears that the best way to establish the best solution for a specific problem distribution is to statistically cross-validate the methods manually (Moore & Lee, 1994; Schaffer, 1993). In some cases, however, as with the management of competing speedup-learning mechanisms, progress has been made when deciding when, where and which speedup mechanism should be selected (Cheng, 1995).<sup>126</sup>

Finally, it is an open question at what level the learning strategies in the toolbox should be modeled. Section 10.1.9 considers whether larger-grain strategies may be modeled with this approach (see also the discussion in Section 9.3). By doing such, Meta-AQUA could better simulate the deliberative goal-driven actions that people choose when learning (e.g., rereading text instructions to clarify information or rehearsing lines to memorize a speech). Conversely, Michalski and Ram (1995) suggest that learning algorithms may actually need more fine-grained decomposition. For example, EBG may be better modeled as a series of knowledge transmutations. At this level, the interactions between transmutations could be as important as they are in the data presented in Chapter IX.

---

126. See Cheng's ISM (Internal Speedup-Mechanism Manager) component of the Theo (Mitchell, Allen, Chalasani, Cheng, Etzioni, Ringuette, & Schlimmer, 1991) general architecture for self-improving problem solvers. ISM is an intelligent agent embedded in the Theo system (i.e., a kind of homunculus) that dynamically manages the selection between caching, EBG, and Bounded-Cost EBG by watching and analyzing the performance of Theo. In effect, the system receives as input a declarative description of the behavior of Theo's architecture, and restructures it to behave more efficiently. This technique is very amenable with the IML approach and may be useful in better understanding the relationship between competing learning mechanisms.

#### 10.1.4 Parallelism in Learning Plans

As far as we know, this research is the only place in which the issue of parallelism in learning has been raised. However, in multistrategy systems that run multiple learning algorithms on a given set of data, the savings in computational resources can be significant if parallel computation is applied. In order to use such an approach, it must be guaranteed that the algorithm running in parallel do not interact. Therefore there must be no data dependencies present so that the order of calls will not affect the results of the learning. This issue is very central to the focus of this research.

Nonlinear plans have the advantage of creating a partial order so that any step not specifically placed before another can be run concurrently. For instance, Tate (1977/1990) discusses the use of non-linear planners to generate *project networks* that formalize a plan as a partially ordered network of actions. Such nets represent steps in a plan, such as the plan used to construct a house. For any given step in the plan, the project network specifies those steps that must come before it and those that must precede it. When constructing the house, the foundation is finished before the walls are installed, which in turn precedes the roof construction. However, the individual steps that insert the wiring and the plumbing into the walls can co-occur, although as sub-steps for the wall construction step, they must follow the foundation. Likewise, a generalized project network, such as those generated by the Nonlin component of Meta-AQUA, specify the steps which may co-occur and therefore can be run simultaneously in a learning plan.

To take advantage of this feature, however, more research needs to be performed to compare the negative effects of learning algorithms in the BK and to formally examine the possible interactions present. Learning operators must then be created for any additional algorithm to enable the planner to create a learning strategy with a partial ordering established. Moreover, the goal taxonomy presented in Section 6.3.1, “Explicit Learning Goals,” must first be more fully developed if progress is to be made in this area.

#### 10.1.5 Learning New IMXPs.

Another question remaining to be examined is whether or not a system such as Meta-AQUA needs to be able to learn new IMXPs or whether a relatively small set of failure patterns can be established to cover most types of common reason failures. One of the benefits of XPs is that they represent abstract patterns that can be adapted and applied to explain many anomalies so that the number required to cover a limited domain is tractable enough to be enumerated by a researcher (Schank, 1986). The open issue with respect to Meta-XP is whether there can be a relatively small set of IMXPs that can sufficiently cover the space of possible failure causes in Table 5, or whether the learning of new causal patterns will be necessary. That is, will the number of required patterns be too large to be exhaustively enu-

merated and can the domain of all reasoning failures in any way be considered “limited?” Are we asking too much of the XP paradigm?

One of the reasons that we did not try to let Meta-AQUA learn new IMXPs is to avoid the possible issue of circular reasoning. If the system uses IMXPs as a basis for deciding what to learn, yet at the same time needs to learn IMXPs, how can the system then have a firm foundation for making any of these decisions? The outcome of considering this question was our decision to see how far the research could proceed without learning IMXPs, and then, if the implementation proved insufficient, to add learning at a later point. Currently, the researcher has the burden of providing all of the IMXP knowledge structures used by the system. In practice, it is still undetermined whether this strategy is a good one. However, until further results show otherwise, this issue need not have a high priority.

#### **10.1.6 Auto-Validation of Learning.**

One of the areas for which very little research has been attempted is the issue of automatic validation of learning results. That is, the system at present does not examine the outcomes of learning in order to establish that the learning was indeed effective (this was the unimplemented step number three of the IML algorithm in Figure 48 on page 126). Although many systems forego this line of questioning altogether, it appears that potential exists for applying the methods developed here toward this end. As mentioned in both sections 5.4.2 and 7.5, the IML algorithm should be able to track the benefits of learning in one of either two ways.

By maintaining a copy of the TMXP associated with the locations in memory that was altered, when the same concepts are re-used in the future, the system could establish whether or not the additions or modifications to the BK are useful in the future. A threshold might be used to estimate whether the new knowledge works by inferring the correctness after a given number of applications of the knowledge succeeds with no additional failure (an example of learning through success). Alternatively, if the changes proves to present more problems than benefits, the trace can be used to roll back the modification. Additional research is necessary, however, to establish the exact details by which such an approach might be justified.

Alternatively, the system could actively test its newly learned knowledge in order to falsify it. If it cannot be easily defeated by a test, then the knowledge can be more firmly believed (yet another instance of learning via success). For example, if instead of trusting the input to the system that dogs bark at containers when detecting contraband, the system might place contraband in a hidden location in order to see if this predicted event actually occurs in additional scenarios. The risk is that side-effects of the plan may negatively affect the reasoner. That is, the test may actually result in the arrest of the agent. Many issues



arise when taking this approach (see Carbonell & Gil, 1990). In order to institute this validation strategy, however, the problem-solving mode of Meta-AQUA must be refined and a better theory of the integration of understanding and problem solving be developed. Although this adds an extra implementational burden, the advantage of doing so results in an additional reasoning strategy. As an additional side-effect, it also furthers the incorporation of other failure causes such as **flawed-heuristic** and **missing-heuristic** which represent failures of choice between multiple problem-solving methods. Thus a contribution to the future research outlined in Section 10.1.1 accrues.

As suggested by Martin (1992), the most interesting possibility for future research is not simply to allow Meta-AQUA to monitor the effectiveness of its learning. Instead, an intriguing possibility is to have a system use the information provided by self-evaluation to decide when an introspective approach is useful. That is, the system may be able to eventually learn the conditions under which introspection is useful itself.

#### **10.1.7 Failure Types as a Cognitive Category**

As mentioned in Section 3.2, failures are of five types: contradiction, impasse, false expectation, surprise, and unexpected success. This categorical division is an implied hypothesis of this dissertation, although it has not been formally tested. It has been given some credence by functional arguments (e.g., that the failure types represent salient symptoms from which the need for learning can be inferred). Also, the computational utility of this category is established because the functionality of Meta-AQUA is based in part on the taxonomy and because Chapter IX demonstrated positive empirical results from the implementation. However, there exists a further claim that these categories have psychological reality.

If this is true, then the hypothesis predicts that human subjects will understand the world based upon these basic category assignments. In speculation, it might be possible that subjects will group stories about various reasoning failures based upon these divisions. An experimental design for such a manipulation might be modeled after the categorization tasks of Wisniewski and Medin (1991).

The taxonomy does not, however, differentiate between failures due to memory retrieval from failures due to inference. As will be remembered from Section 4.7, the representations of these failure types possess **Cognize** nodes that can be refined as either a memory process or an inferential process. Therefore, the taxonomy representation predicts that, for instance, people will group both impasse due to forgetting along with impasses due to not being able to infer a conclusion. Although such speculation is premature, this or a similar psychological experiments might further validate IML theory or provide a reason for changing it to a more plausible state.

### 10.1.8 Extending the Computational Evaluation

The current section explores the further validation required to substantiate hypothesis number one as reported in Section 9.2. The first subsection describes the research needed to finish the study already begun. It adds two more empirical manipulations to the one already present in the experimental design. The second subsection explains how the conditions under which the hypothesis holds can be established more rigorously.

#### 10.1.8.1 Establishing the hypothesis that introspection facilitates learning

As mentioned in Section 9.1.1.2, to fully test the hypothesis that introspection facilitates learning, a more complete set of manipulations should be performed. If both blame-assignment and deciding what to learn have introspective components, then to pinpoint the effectiveness of introspection, both phases should be systematically ablated and evaluated for effects to the performance task. The mapping from symptoms of failure to selection and ordering of repairs (i.e., learning-strategy construction) is as shown in (10), whereby blame-assignment produces the causes of failure and deciding what to learn produces the learning goals.

$$\text{symptoms} \rightarrow \text{causes} \rightarrow \text{learning-goals} \rightarrow \text{repairs} \quad (10)$$

Chapter IX evaluated learning without the mediation of learning goals, but some of the power of the introspective method may actually be centered in either the blame-assignment phase, or in the interaction between blame-assignment and deciding what to learn. Therefore, a full evaluation must also run the same experimental conditions as described in Section 9.2.1, without the causal determination from blame-assignment (i.e., using a mapping from failure symptoms directly to learning goals) and without the use of either blame-assignment or deciding what to learn (i.e., using a direct mapping from symptoms to repair). The inclusion of these two additional experimental manipulations can better isolate the causal functions of introspection.

#### 10.1.8.2 Determining the conditions under which the hypothesis holds

To further investigate the conditions under which the IML method is best suited, the initial conditions of the program and the data should be varied. That is, both the program state in terms of parameter settings and the state of the BK was held constant across runs in all three conditions of the independent variable. The parameter settings of the Tale-Spin program that provided the input was also held constant. By varying these conditions it can be established under what conditions the hypothesis that introspection facilitates learning holds. It is not expected that under all conditions introspection outperforms a more reflexive approach because of the considerable computational overhead of the IML method.

Moreover, to further establish the generality of the IML method, these trials should be run with a different performance task than story understanding. A preliminary problem-solving mode for Meta-AQUA has already been developed. However, as mentioned in Section 10.1.6, the creation of a robust problem solver in the IML framework will be nontrivial. Not only must additional code be developed, but a general theory of the interaction between problem solving and learning should be developed if the implementation is to be successful.

### 10.1.9 Extending the Model of Human Learning

As described in Section 9.3, Meta-AQUA has been modified to simulate novices that learn to program in LISP. Pirolli and Recker (1994) produced a set of protocols from human subjects in an experimental setting that had an especially relevant feature. Between trials the subjects would spontaneously generate protocols concerning their own understanding of the problems previously solved. The study reported that those subjects who exhibited metacognitive explanations (i.e., explanations about their own comprehension and their own failures) tended to learn better than those who did not. We chose to simulate a fragment of a particular subject (AK88) because it combined three different strategy statements and so represents an instance most like the learning-strategy construction task the Meta-AQUA program models.

The result of the simulation was favorable. Meta-AQUA modeled most of the protocol sufficiently, both from the subjects cognitive task (problem-solving) and the subject's meta-cognitive task (strategy construction). However, because the problem-solving mode of Meta-AQUA is not fully implemented the simulation of the programming task was overly simplified. And because the system's memory module is incomplete, the section of the protocol pertaining to the subject's reminders of a previous problem solving episode (i.e., a programming problem) was not finished. Only the first third of the segment was fully simulated. Therefore, to finish this trial with Meta-AQUA, the system must be programmed with both a more robust problem-solving process and a more plausible memory. Previous sections discuss the need for an integration of problem solving with understanding. Finally, a number of additional protocols need to be further simulated for IML theory to be accepted as a sufficient model of human metacognitive behavior.

### 10.1.10 Extending the Representational Vocabulary

Chapter IV described in detail the representation language with which Meta-XPs are represented. The focus of the chapter was to develop a representation for reasoning failures and to show how they can be composed from a primitive vocabulary of terms that constitute an ontology of mental states and actions (see Figure 19 on page 70). The utility of a complete representation of mental terms has a number of advantages. Not only can a declaratively represented mental vocabulary enhance learning and reasoning about the self, but it

can also enhance a system's ability to understanding the mental properties and knowledge of other agents. Not only in artificial systems but also in human reasoners, the ability to understanding the mental world of others is important (Jameson, Nelson, Leonesio, & Narens, 1993). This section simply notes that to finish the desired ontological taxonomy many more mental terms need representation.

A goal of this research is to derive a formalism that can express states and mechanisms of the mental world. A comprehensive representation needs to be delivered that can be shared between existing formalisms, and where the reuse of representations facilitate both uniformity and transfer across domains in order to support intelligent reasoning, understanding and learning. In support of these goals, this section lists a number of concepts and dimensions that demand representation and remaining issues that must be considered in the pursuit of these research goals. Although it is perhaps premature to speculate on the exact means with which they can all be represented, the following list presents a provocative enumeration of potential candidates.

1. **introspect, retrospect, inspect, reflect, suspect, expect.**
2. **expect** versus **hope** versus **wish** - depends on knowledge or certainty, that is, one may expect something when confident, but hope for an outcome when doubting, and finally wish for something that is least likely to occur. I suppose we **pray** for the impossible; which leads this list to item 3.
3. **wishful thinking** (in the face of conflicting evidence along with rigid beliefs).
4. **suspend thinking, resume thinking** (both of which are opportunistic), **reconsider** (in the light of hindsight, rather than after an interruption usually).
5. **suspend belief, (day)dream, imagine, wonder.** See Schank et al. (1972) p. 18, for a crude representation of wonder; p. 30 for imagine).
6. **apprehension, perception, precept, percept.**
7. **foresee** (foresight), **review** (hindsight), **intuit** (insight).
8. Under foresight: **foretell, apprehension, wish, hope, and expect.**
9. Under hindsight: **review, retrospect, remember, reconsider, reflect.**

10. Under insight: **premonition, presentiment, apprehension, hunch, discovery, pre-science**.
11. **algorithmic** versus **heuristic** processing, e.g., projective reasoning via possible worlds given different assumptions (logic) versus given different interpretations (analogy or metaphor).
12. **group** (categorize), **compare, contrast, analogize**.
13. **want, need, desire, goal possession** (all items equivalent).
14. **rehearse, elaborate, search; baffled, stumped, perplexed**.
15. **notice, observe, surveil** (search), **discover, await** (depends on target or expectation; another scale by deliberation and knowledge explicitness).
16. **intend** (to do an *act*) versus **goal** (to achieve a *state*).
17. **intend, attend, pretend, suspend, portend, comprehend** (understand). See Schank et al. (1972) pp. 70-75, for an early discussion illuminating the often **incomprehensible** mental term of understand ;-)
18. **(self)explain**.

## 10.2 Learning Bias and Category Membership: An extension

Cox and Ram (1994b) have argued in previous work (see also Appendix A, “THE DEGREES OF FREEDOM IN LEARNING”) that failure provides a computationally efficient bias-filter for input examples in machine-learning systems. Results in cognitive science (e.g., Chinn & Brewer, 1993) have likewise demonstrated a complementary role for anomalous data in revising background knowledge in scientific and naïve theories. There are two major reasons that failure is a good bias from which to learn in both machines and humans. Failure guarantees that something worth learning exists, and it also guarantees that the degrees of freedom in learning are less than those when learning from success. A novel research extension exists from which to apply this result to concept learning. This extension investigates the interaction of failure and knowledge during categorization tasks.

Typical theory-revision systems contain a single-concept background theory. For example, a system that contains the classical cup theory (e.g., EITHER, Mooney & Ourston, 1994) assigns either “cup” or “non-cup” to all input examples and then adjusts its domain rules when errors occur. Solitary classification systems, however, are not cognitively plausible. People do not fail simply by classifying a cup as a non-cup. Instead there exists a false-assignment category (such as bowl) that competes with the correct category. Failures then often lead human learners to use a “compare and contrast” procedure by which knowledge of the categories is refined. Yet not only is it significant that people perform this procedure, and thus such algorithms are worth discovering, but since more constraints exist under which learning can take place, such algorithms may be computationally much more tractable. This dictates that category learning should take place in the context of multi-category theories (Mooney & Ourston, 1991, report related progress in this area).

Using Meta-XP structures, IML theory has declaratively represented a number of reasoning failures that Meta-AQUA can reason about explicitly. The typical reasoning failure in category assignment is a failing positive (Mooney, 1993) such that a theory falsely categorizes an example,  $x$ , as a member of an expected category,  $E$ . Instead,  $x$  should be categorized by some other theory as a positive member of the actual category,  $A$ . Meta-AQUA uses a Meta-XP called an *expectation failure* to reason about such situations in story-understanding tasks (i.e., the contradiction failure type). During mis-classification in a multi-category domain theory, it is guaranteed that the category  $E$  is overly general and the category  $A$  is overly-specific. Thus, for propositional Horn-clause theories, an extra rule or missing rule-antecedent must exist in  $E$  *and* an extra rule-antecedent or missing rule also exists in  $A$ .

Although not enough room exists here for a detailed description, some preliminary heuristics for taking advantage of these constraints have been established. Moreover, during mis-classification, an agent should consider not just the fact that it thought  $x$  was a member of  $E$ , but was not; rather, the learner should also consider why  $x$  was a member of  $A$ . The agent can then compare and contrast the concepts  $E$  and  $A$ , the reasons why  $x$  was thought to be a member of  $E$  with the reason why  $x$  was thought *not* to be a member of  $A$  (if that was considered at all), and if  $A$  was not considered, then why not (was a memory association incorrect?). Many theory-revision systems compare  $x$  only with the theory supporting  $E$  and cannot search for errors that may be related to an interaction between multiple categories; furthermore, none support memory errors as does the research present here.

Finally, although a failing positive implicitly implies a failing negative (or perhaps a novel category), the inverse does not necessarily hold. When a successful negative occurs, the judgement may result in either a successful positive or an impasse. That is, it may know that the example is not a cup, but may or may not know what the actual category is. Meta-XPs can represent both related cases as either a forgotten category (*missing-association*) or as a novel category (*novel-situation*). Neither case has been treated by current category

theory-revision systems. Although future research on this issue can make contributions to concept revision systems based on propositional Horn-clause logic, the intention is to go beyond such formulations to include hierarchically-structured case memories. Experience with case-based reasoning and explanation-pattern theory will facilitate such further extensions of concept learning through knowledge-intensive explanation and reflection (e.g., comprehension monitoring).

A question often raised when watching demonstrations of Meta-AQUA refine conceptual categories, such as its knowledge of dog-barking, is how the system determines that the reason the dog barks is that the dog detects *some* amount of contraband inside the luggage, rather than the fact that it detects exactly two kilograms of the contraband. The answer is that it possesses an explanation about authorities who detect explosives in a previous smuggling story, and thus the system can adapt that explanation to constrain the inferences that occur during comprehension. It is the contraband, rather than the amount of contraband, that is the focus of the explanation. Although superficially similar to the drug-bust story, HC1, a much more difficult passage for a system to understand would be the story of a dog who learns to perform tricks (see Figure 99).

S1:A dog was trained to appear to count.

S2:When the trainer held up various objects, it would bark appropriately.

S3:The dog barked twice because it detected two bowling pins in his hand.

---

Figure 99. The trick story

In this case, the number of objects that the trainer holds, rather than the kind of objects, is crucial to the explanation. The role of knowledge and explanation is much more complex in understanding this story despite the shorter length. In particular, one cannot always depend on having a past case to adapt when explaining a story. If the research goals outlined in this section are accomplished, IML theory will be much closer to accounting for such *ad hoc* categories (Barsalou, 1983) as *barking based on the number of objects*. This will support the overall goal of establishing a more complete theory of multistrategy learning from both cognitive science and machine learning perspectives.

### 10.3 Understanding student explanation failures: An application

The goal of this applied research is to develop methods by which intelligent learning environments can automatically detect student explanation failure and can provide appropriate feedback so that the student can correct the mistakes. In order to succeed at this goal,

it is necessary to be able to declaratively represent typical explanations that students make and to be able to reason about the content and structure of these explanations. The problem distills to that of mapping from symptoms of explanation failure to causal patterns that represent why the failure occurs, a task central to IML theory. Once a system understands the reason an explanation is wrong, it can then provide accurate feedback to the student so that better explanations are generated in the future. By doing so, the student will be able to more adequately understand the target domain in which learning takes place.

In order to develop intelligent learning environments that are effective in training, it is essential to monitor student progress so as to be able to provide proper feedback. To create a student model, many systems use some variant of template matching in order to infer the propositions or declarative facts the student knows. This information can then be used to provide missing knowledge or to correct erroneous knowledge. However, student inadequacies often are not limited to such interpretations. Isolated rote facts are not the only factors that lead to poor understanding; rather, a student's performance is more accurately estimated by how well they can explain a target domain during troubleshooting or problem solving. The goal of this research is to develop and explore computational methods whereby student explanations can be automatically evaluated and debugged by an intelligent system in order to assist the student to form better explanations on their own.

Schank, Fano, Bell, & Jona (1993) support the principle that effective teaching systems must present goal-based scenarios. That is, rather than simply absorbing information that is passively presented, the student must possess specific goals that motivate and direct the learning process. Therefore, a system that integrates problems solving or troubleshooting provides a more effective learning environment than do traditional browsing or tutoring systems. Moreover, not only are goals useful, but the student should generate explanations in support of these goals. Chi and her colleagues (Chi et al., 1989) have demonstrated that students who spontaneously generate explanations about a given domain perform better and learn more thoroughly than do students who generate less "self-explanations." This increase in performance is not gained by providing the explanations to the student, but is found when the student can generate them independently. Hale and Barsalou (1995) have shown that a distinct difference exists between the types and effectiveness of explanations generated during the troubleshooting of physical systems than when students are learning about the system and the facts related to the system. The quality and content of the explanations generated by students differ depending on the goal of the student. Also, Pirolli and Recker (1994) have reported that students who reflectively debug their own reasoning errors and understanding tend to be the better learners.

Given such results, the target of this line of future research is to develop environments that support student explanations during troubleshooting in key skills such as engine diagnosis. The troubleshooting task for the student is to map symptoms of system failure to faults in the system (e.g., an aircraft engine system). Explanations in this task consist of



providing justifications or causal linkages for why a given fault leads to a given symptom.<sup>127</sup> An intelligent learning environment that possesses a capability to improve student explanations during troubleshooting must have the following major components:

1. A mechanism to represent and detect failed student explanations.
2. A procedure for explaining why student explanations fail.
3. A method for deciding what needs to be learned (by the student).
4. A delivery system for providing the feedback to the student by asking questions that prompt the student to re-explain and to discover the student's own errors.

With these four components, learning will be enhanced by supporting, not only the principles and facts contained within a target domain, but the student's ability to explain the relations and dynamic interactions within a domain, and also by supporting the student's own comprehension monitoring. By emphasizing troubleshooting, the student will understand how target systems operate in all conditions, not just a surface understanding of normal operating conditions. The four step procedure above is a direct adaptation of the research presented here on introspective explanation of reasoning failures, and therefore represents an application of well-developed theories and computational algorithms.

To effectively build the first component above, a system must be able to represent student explanations in a declarative format. The explanation pattern (XP) knowledge formalism (Leake, 1992; Ram, 1994; Schank, 1986) presents an ideal representation that has been tested in a number of domains. In support of the input side of this component, a substantial natural language and graphical interface must be constructed to translate student expressions into XP format. A theory of explanation failure detection has already been worked out (Leake, 1992), although work remains to further generalize it.

The second component of the system is the blame assignment task. The purpose of this blame assignment task is to map symptoms of student explanation failure to explanations of the causes of failure, not unlike the student's task of mapping symptoms of device failure to failure causes. The result is an explanation of an explanation failure. The Meta-XP knowledge structures directly support such explanations. As explained in Chapter IV, these structures represent abstract causal patterns of reasoning failure. In support of this data structure, Chapter III created a comprehensive taxonomy of failure causes. This tax-

---

127. Note that this is device blame-assignment by the student.

onomy is both domain independent and task independent, thus making it well suited for the student troubleshooting task.

Algorithms involved in the third proposed component have been described in Chapters V through VII. The task is to take an explanation of reasoning failure and to generate a set of partially ordered learning goals that, if achieved, will reduce the likelihood of the failure from being repeated. Although this work has concentrated in system self-evaluation, the transformation to an evaluation of the requisite learning of an external agent should be tractable. Work will be required to fully elaborate a set of suitable learning goals in this new domain. The theory has already been applied, however, to the domain of troubleshooting by human operators in an electronics assembly plant (Ram, Narayanan, & Cox, 1995), but has never been applied to debugging explanation of actual troubleshooters in a learning environment.

Finally, rather than provide a corrected explanation directly, the goal of the fourth component is to allow students to debug their own explanations. Once the system diagnoses the student's explanation failure, a series of increasingly specific hints and examples can be provided to the student that target the failure in light of the inferred student's learning goals (Hume, Michael, Rovick & Evens, 1996). These are presented in sequence until the student is able to explain the cause of the failure and provide the proper explanation. Much research will be involved to refine an effective way of performing this task.

As previously noted, missing and erroneous knowledge is not the only source of reasoning failure. Students errors may also occur due to memory organization problems. That is, a piece of knowledge may be present, but the associations connected with such items in memory may be missing or flawed. As an example from Barsalou, Hale and Cox (1989), consider the types of rules a student engaged in troubleshooting four-stroke engines may have learned. Two possible learned rules are that when a strong smell of gas exists during engine operation, the choke may be broken and that when the engine is manufactured by Briggs and Stratton, the condenser may be broken. However, if a strong smell of gas in a Briggs and Stratton engine is indicated, the student may try to test the condenser rather than the choke. The problem is not with the students knowledge; rather, the problem is with the memory associations connected to the rules. A system might infer this causal factor and attempt to provide an example that describes the fuel system. Chapter III provides additional examples of causal factors implicated in explanation.

This work is novel because previous student modeling has not included an analysis of student explanation and self-comprehension. Although such an application of IML theory is ambitious, the chances of success are high because the new research has a firm foundation in the results produced by this dissertation. The research promises to add an intelligent dimension to training systems that support effective learning, especially in the crucial area of explanation formation and troubleshooting skill expertise.

## 10.4 Summary

This chapter discussed future research in terms of open issues with the current state of IML theory, extensions to the theory, and avenues for application. Because the topics of this dissertation are so broad and wide-ranging, many directions exist to further the research. We began by enumerating and explaining ten issues remaining underexplored and directly stemming from this work and described the approaches that can be taken toward them in the future. The chapter then outlined two independent research directions that depend upon IML theory. First, the issue of learning bias and category formation extends the reach of IML theory into the learning of categories using a compare and contrast heuristic. This heuristic is cognitively inspired from the behavior of humans, but applied to the domain of Horn-clause logics. Secondly, we suggest building intelligent tutoring systems that understand student explanation failures in the same manner that Meta-AQUA understands its own explanation failures. These three categories of future research efforts (open issues, extensions and applications) demonstrate that IML theory has significant potential for generating new inquiry.



## CHAPTER XI

### RELATED RESEARCH

*Here hills and vales, the woodland and the plain,  
Here earth and water seem to strive again,  
Not chaos-like together crush'd and bruis'd,  
But, as the world, harmoniously confus'd:  
Where order in variety we see,  
And where, though all things differ, all agree.*

—Alexander Pope (1713).

Artificial intelligence certainly does not have a monopoly of interest concerning introspection and related topics such as learning and multistrategy reasoning. Philosophers and observers of the human condition have been fascinated by the subjects for a very long time. Around the turn of the 16th century in *De Trinitate*, Augustine asks “What then can be the purport of the injunction, Know thyself? I suppose it is that the mind should reflect upon itself.”<sup>128</sup> More recently, Hofstadter (1979/1989) convincingly argues that the concept of reflection, or an object turning in upon itself (i.e., his concept of “Strange Loops”), is a common and powerful theme, in and outside of science. Strange Loops can be found in mathematics (with the proofs of Gödel), art (with the painting of Escher), and music (with the compositions of Bach). But with few exceptions (e.g., Lyons, 1986, Pollock, 1989a), AI and cognitive psychology present the only mechanistic explanations for such phenomena and represent the only disciplines that address the issue in the context of learning.

The research that relates to the issues presented in this dissertation concerning multi-strategy learning and introspection is prolific, and we have mentioned a number of the related research works already. As a multistrategy learning theory, we have shown that the ideas behind Meta-AQUA fit into the strategy selection category of multistrategy models (see Section 5.4.1). In terms of failure-driven learning, we have characterized IML theory

---

128. Cited in Lyons (1986, p. 1).

as a loose coupling between blame assignment and repair, in direct contrast to those theories that mandate a tight coupling (see Section 7.1). Because the preceding chapters have already connected much of the AI and psychology literatures to these and other individual issues, this chapter will not replicate the discussions. Instead, the chapter examines the contrasting (and complementary) approaches to reasoning about the mental domain both literatures report, particularly in the context of learning. It will review the history and current status of research into introspection of mental processes and reasoning about knowledge or beliefs. Partitioned into two parts, this chapter will examine the broader issues that relate to introspection and introspective learning from an AI perspective (Section 11.1) and from a psychological perspective (Section 11.2). The chapter concludes with a brief summary and discussion (Section 11.3).

## 11.1 Artificial Intelligence, Metareasoning, and Introspective Learning

The AI community has long considered the possibility of providing machines with reflective faculties. In recent years, a number of conferences and symposia have been organized to explore some of the issues that relate to this concern: the Workshop on Meta-level Architectures and Reflection held in Alghero, Italy, during October, 1986 (Maes & Nardi, 1988); the International Workshop on Machine Learning, Meta-Reasoning and Logics held in Sesimbra, Portugal, during February, 1988 (Brazdil & Konolige, 1990); the IMSA-92 Workshop on Reflection and Metalevel Architectures held in Tokyo, Japan, during November, 1992; the AAAI Spring Symposium on Representing Mental States held at Stanford University during March, 1993 (Horty and Shoham, 1993); and the AAAI Spring Symposium on Representing Mental States and Mechanisms held at Stanford during March, 1995 (Cox & Freed, 1995). In general, the loci of related research efforts has tended to focus the logic community on belief representation and introspective reasoning about such beliefs; the expert system community on metaknowledge and the control of rules; the planning community on search control and the choice of reasoning actions; and the case-based reasoning community on reasoning about reasoning failure and representations of process. This section reviews the varieties of analysis pertaining to a computational theory of introspection and, more specifically, of introspective multistrategy learning.

From the very early days of AI, researchers have been concerned with the issues of machine self-knowledge and introspective capabilities. Two pioneering researchers, Marvin Minsky and John McCarthy, pondered these issues and put them to paper in the mid-to-late 1950's. Although first exchanged among colleagues, and then printed at conferences at the turn of the decade in preliminary form,<sup>129</sup> reprints of these papers were refined and gathered together in the seminal collection of early AI articles entitled *Semantic information processing* (Minsky, 1968b). Minsky's (1968a) contention was that for a machine to adequately answer questions about the world, including questions about itself in the world, it would have to have a executable model of itself. McCarthy (1968) asserted that for a

machine to adequately behave intelligently it must declaratively represent its knowledge, including knowledge of itself. These two positions have had far-reaching impact.

Minsky's proposal was procedural in nature while McCarthy's was declarative. Minsky believed that an intelligent machine must have a computational model of the outside world from which a simulated execution could answer questions about actions in the world without actually performing any action. He argued that if a machine uses models to answer questions about events in the world and the machine itself is in the world, then it must also use a recursive self-model or simulation to answer questions about itself, its own dispositions, and its own behavior in the world. This was a very early prototype of a mental model that became a precursor to similar research in both problem solving and understanding (e.g., Bhatta, 1995; Bhatta & Goel, 1992; Johnson-Laird, 1983;<sup>130</sup> deKleer & Brown, 1983/1988; McNamara, Miller & Bransford, 1991). In the spirit of Minsky's original theme, some very novel work has also been performed to enable a machine to procedurally simulate itself (e.g., Stein and Barnden, 1995).

Although a closer examination of Minsky's propositions will be deferred until Section 13.2 of the epilogue, Section 11.1.1 explores McCarthy's proposals and their local impact on the logic community and their more global effect on the tone of research into a computational explanation of introspection. Section 11.1.2 then looks at additional varieties of research in the expert-system and planning communities. Finally, Section 11.1.3 relates some of the relevant research from the case-based reasoning and multistrategy learning communities to the research presented here.

### 11.1.1 Logic and Belief Introspection

As mentioned above, McCarthy (1968) not only establishes a manifesto for AI (i.e., knowledge representation is foundational, especially in declarative form), but suggests that machines can examine their own beliefs when such beliefs are explicitly represented. This suggestion is developed in McCarthy and Hayes (1969) and made explicit in both Hayes

---

129. Minsky notes that he had been considering the ideas in this paper since 1954. It first appeared as Minsky (1965), although the concluding two pages of Minsky (1961/1963) address exactly the same issue. A significant portion of McCarthy's ideas was first published as McCarthy (1959).

130. Johnson-Laird (1988, p. 361) explicitly takes issue with the suggestion that Minsky's concept of a self-model was in such a form that it could correspond to a human's capacity for self-reflection. He claims that Minsky's formulation is equivalent to a Turing machine with an interpreter that consults a complete description of itself (presumably without being able to understand itself), whereas humans consult an imperfect and incomplete mental model that is somehow qualitatively different. However, this argument appears to be extremely weak because the two positions are so similar and closely related.

(1979/1981) and McCarthy (1979). A system requires such an introspective capability if it is to reason fully about the correctness of its knowledge. This is especially useful because beliefs are subject to retraction in the face of new information (i.e., knowledge is nonmonotonic). But beyond any technical details, McCarthy also wonders what it means for a machine to have a mental life. McCarthy (1979) enumerates six reasons why attributing mental qualities to programs and machines is a useful effort. Among them, he claims (as does Dennett's 1978 essay on the *intentional stance*) that humans can more quickly and more easily understand a program, its behavior, and its intended function by ascribing beliefs and goals to the machine than by analyzing and explaining it in the language of program code and computer states. But most interestingly, McCarthy takes the business of understanding and simulating a machine's mental life beyond a mere practical metaphor. He questions what it means for a machine to have consciousness and to introspect about its mental world. Furthermore, he realizes that "introspection is essential for human level intelligence and not a mere epiphenomenon." (McCarthy, 1995, p. 89) Thus, he is keenly interested in the relation between machine and human introspection.

McCarthy (1979) defines introspection as a machine having a belief about its own mental states rather than about propositions concerning the world. This position has focussed the logic community, especially researchers such as Konolige (1985, 1988) and Moore (1995), on reasoning about knowledge, belief, and internal states, rather than reasoning about process and computation. McCarthy (1993) further formalizes the idea of introspection by introducing context as a first-class object that can be reasoned about. By encapsulating mental situations in formalized contexts, the reasoner can view the mental state as providing an outer context. Reasoning about one's own thoughts then involves transcending the outer context (McCarthy, 1993). Unlike our work, however, the realization of such an introspective mechanism has not been implemented in a computational system. Furthermore, McCarthy (1995) notes that even though reason maintenance systems (e.g., Doyle, 1979) record justifications for their beliefs and can retract beliefs in response to new information, they do not have the capability of inspecting the justification structures or making specific assertions about them, nor do they have the power to derive explanations from such structures.

Although some in the logic community have generalized the logical approach to cover many mental attitudes (e.g., Ballim, 1993),<sup>131</sup> the preponderance of research has been limited to deductive reasoning concerning belief and uncertainty. This same research bias is evident in the philosophical literature on mental representations of internal states and the

---

131. McCarthy (1979; 1995) also outlines a number of additional issues concerning the mental domain that have received lesser attention by the logic community. He raises the issue of consciousness, language, intentions, free will, understanding and creativity, all of which have come to represent provocative focal aspects of intelligent reasoning.



theory of mind (e.g., Stich & Warfield, 1994; but see Lyons, 1986, for a prominent exception). Because of the philosophical roots of logic, much of the research by logic-oriented researchers tend to focus on *justified true belief* and how one can *know* a proposition is true (classical epistemology). The focus is on idealized thought, rather than on developing descriptive theories of actual human error-prone thought. Logical theories are prescriptive in the sense that they outline how people ought to think, not how they do think. This research focus has forced the logicians into grappling with how reasoners can maintain a consistent and complete set of propositions and how reasoners can deduce only correct propositions from such a set. The advantage of such an orientation is that a system can have confidence about inferences and propositions in this set; but, the disadvantage is that if the set becomes inconsistent, then from one inconsistency a system can prove false assertions. Despite the fact that McCarthy and others have imbued research on introspection with a sense of legitimacy, the most damaging general criticism of the logic approach is that the metaphor of “thought as a logical proof” is limited when applied to human thinking and learning. Much of the reasoning process is simply not deductive (McDermott, 1987/1992). But with respect to our IML theory, a critical omission is that learning is never addressed by their intellectual dialogues.

### 11.1.2 Knowledge-Based Systems, Quasi-Introspection, and Related Theories

The expert system community has also invested much effort into the formalization of metareasoning and metaknowledge. It was recognized in the late 1970’s that differences exist between domain knowledge in the form of expert rules, and declarative control knowledge in the form of meta-rules (Davis, 1979, 1980; see also Clancey & Bock, 1985). Metarules encode knowledge about how rules should be executed, whereas ordinary rules encode domain-specific knowledge. Barr (1977, 1979) noted, as we do here, the parallel relation between higher-order knowledge and reasoning by knowledge-based systems and human metacognition (see also Lenat, Davis, Doyle, Genesereth, Goldstein, & Schrobe, 1983). Especially when trying to automate the transfer of domain knowledge from human expert to machine expert, these and other researchers have attempted to give programs abstract knowledge of human reasoning and inference procedures, so that programs can understand human experts (see for example, Clancey, 1987). Additionally, when expert systems explain a conclusion by providing to the user a list of rules through which the system chained to generate the conclusion, the system is said to introspect about its own reasoning. This view appears, however, to be an over-simplified example of both introspection and explanation.

Davis and Buchanan (1977) claim that four types of meta-level knowledge exist: knowledge about object representations (encoded in schemata), knowledge about function representation (encoded in function templates), knowledge about inference rules (encoded in rule models), and knowledge about reasoning strategies (encoded in metarules). But much of this information is less akin to introspective knowledge than it is to ordinary

abstract knowledge. For example, to claim that default inheritance and learning are inherently introspective processes (Maes, 1987b) or that extrapolating from past experience is reflective thinking (Smith, 1982/1985) is perhaps stretching the definitions of introspection and reflection, respectively.

As another example, Genesereth (1983; also Maes, 1988) considers the meta-level to be that which decides about the base-level (or actions in the world) and explicitly includes planning as a meta-level reasoning process. This unfortunately conflates metareasoning with reasoning because the system is not reasoning about the reasoning process itself.<sup>132</sup> Instead, three levels exist: object, reasoning, and meta-reasoning levels. For example, John Self (1992) argues that a metacognitive component is crucial in student modeling for intelligent learning environments and proposes three levels. The base level, B, contains both rules and propositions specific to the particular tutoring domain. The reasoning level, R, contains descriptions of the processes that operate on and change the B level. Finally, the meta level, M, contains descriptions of those processes that monitor the progress of the reasoning level processes and reason about the outcomes of the R level. Processes in the R level produce changes in the B level, and processes in the M level produce changes in the R level. Stefik (1981) also emphasizes this three-level configuration.

Another popular research issue is to develop systems that can reason about LISP functions and the actual code that represents a program's control (Batali, 1983; Davis & Buchanan, 1977; Maes, 1987a, 1988; Smith, 1982/1985). However, this form of "introspection" is too low-level. Programs need to reason about the functioning at the level of cognitive process, not at the level of program execution.<sup>133</sup> Some in the AI community are recognizing some of the more subtle differences between the different families of metareasoning. For example, Clancey (1992) notes that many of the metarules employed by systems such as TEIRESIAS (Davis, 1979), although dealing with control, are nonetheless domain specific. He claims that strategic knowledge is inherently procedural whereas domain specific knowledge is rule-based. Moreover, unlike his previous work (e.g., Clancey, 1987), he currently eschews modeling the mental process that the expert uses when reasoning about the domain, and instead emphasizes modeling the domain that the expert knows. This change

---

132. A procedural difference exists between reasoning about a solution or a problem and the metareasoning directed at the reasoning that produces such solutions or engages such problems. For instance, Carbonell (1986) notes that in order to transfer knowledge from programming a quick-sort problem on a computer in Pascal to solving the same problem in LISP, a student cannot analogically map the Pascal solution to LISP code. The languages are too dissimilar in data structures and process control. Instead the reasoner must reason about how the original solution was derived and what decisions were made while solving the first problem, analogically mapping the derivation to LISP. Reasoning is at the algorithm level, rather than the code level.

133. In the terms of Newell (1982), the reasoning should be at the symbol level rather than at the register-transfer level of intelligent systems.

of focus, however, seems to be as much a concession to the difficulty of representing introspective knowledge as it is a necessity dictated by representation itself.

Although many in the artificial intelligence community have recognized the necessity of reasoning about one's own beliefs, few have both modeled and represented the processes that *generates* beliefs, and made them available to the reasoner itself. In this category of reflective systems, a distinction exists between those systems that reason forward to decide what action to perform or what computation to execute, and those that reason backward to explain a failure or learn. This is related to the distinction made in the psychological literature between forward strategic control and backward metacognitive monitoring (see the discussion of Nelson & Narens, 1990/1992 in Section 11.2.3). In the former category, systems attempt to choose a reasoning action based on some knowledge of the mental actions at the disposal of the system. Doyle (1980), as well as Russell and Wefald (1991a, 1991b; Tash & Russell, 1994), use probabilistic estimations to decide which computation has the most likely utility. Etzioni (1991) uses decision-analytic methods to weigh the trade-off between deliberation cost, execution cost and goal value when choosing a goal toward which to direct attention and when deciding which action to take in service of a chosen goal. The latter category of systems (backward metacognitive monitoring) is examined in the next section.

By some measures, few people are working on introspection, but in another sense used by some in the AI community, everyone in AI must be working on introspection and metareasoning. Most intelligent programs deliberate to some extent about the types of actions that are optimal given their goals. For example, Soar (Newell, 1990; Laird et al., 1986; Rosenbloom et al., 1993), Theo (Mitchell, et al., 1991), and PRODIGY (Carbonell et al., 1991; Minton et al., 1987) are all programs that make deliberate decisions as to the best action available in their domains. Moreover, if metaknowledge is taken to be any abstract knowledge (e.g., default knowledge), and metareasoning is any of the higher cognitive functions (e.g., planning), then ironically CBR is one of the few non-introspective reasoning paradigms remaining because it reasons from concrete experience and episodes (tokens, rather than abstract types). We agree with Maes' (1987b) assessment that an introspective system is one whose domain is itself. But in the research presented here, we further define an introspective reasoner to be a system that *reasons* specifically about itself (its knowledge, beliefs, and its reasoning process), not those that simply *use* such knowledge. As demonstrated in Chapter IX, this is crucial for effective learning.

### 11.1.3 Relation of AI Research to IML Theory

Although many systems use higher order knowledge and processes in intelligent tasks, the theory presented here is one of only a few that use introspection to support learning in response to failure. That is, the Meta-AQUA system keeps a trace of its reasoning in order to reason backwards towards the failure causes and to formulate a plan to change its

knowledge. As reiterated in this text, several fundamental problems must be addressed to create such learning plans or strategies. These problems are (1) determining the cause of a reasoning failure (blame assignment), (2) deciding what to learn (learning goal formulation), and (3) selecting and ordering the best learning methods to pursue its learning goals (strategy construction). A learning system that determines why reasoning fails, formulates its own learning goals, and constructs a learning strategy needs the ability to introspect about its own reasoning processes and knowledge. In support of such an assertion, Collins et al. (1993) argue that to plan effectively a system must have an explicit model of its of planning and execution processes.<sup>134</sup> In this dissertation, we have argued more generally that to learn, a system must have an explicit model of its performance task.

In general, our orientation is similar to the approaches based on reasoning traces (e.g., Carbonell, 1986; Minton, 1988; Sussman, 1975) or justification structures (e.g., Birnbaum, Collins, Freed, & Krulwich, 1990; deKleer, Doyle, Steele, & Sussman, 1977; Doyle, 1979) to represent problem-solving performance and to other approaches that use characterizations of reasoning failures for blame assignment and multistrategy learning (e.g., Kass, 1990; Mooney & Ourston, 1991; Owens, 1990a; Park & Wilkins, 1990; Redmond, 1992; Stroulia & Goel, 1992). Reasoning trace information has primarily been used for blame assignment during planning (e.g., Collins et al., 1993; Birnbaum et al., 1990; Veloso & Carbonell, 1994) and for speedup learning (e.g., Mitchell, Keller, & Kedar-Cabelli, 1986). A major difference between our approach and these approaches is our use of explicit representational structures (Introspective Meta-XP's) to represent classes of learning situations along with the types of learning needed in those situations.

Other types of knowledge may also be important in multistrategy or introspective learning systems. For example, Pazzani's (1991) OCCAM system has generalized knowledge about physical causality that is used to guide multistrategy learning. In contrast, we propose specific knowledge about classes of learning situations that can be used to guide learning strategy selection and construction. The IULIAN system of Oehlmann, Edwards and Sleeman (1995) maintains metacognitive knowledge in declarative introspection plans. The RFermi system (Kennedy, 1995) maintains goal and memory search information to represent knowledge about its memory performance. This information allows it to introspectively determine improvements in its search behavior. Freed's RAPTER system (Cox & Freed, 1994; Freed & Collins, 1994) uses three types of self-knowledge when learning. Records of variable bindings maintain an implicit trace of system performance, justification structures provide the knowledge of the kinds of cognitive states and events needed to explain the system's behavior, and transformation rules (Collins, 1987; Hammond, 1989) describe how the mostly implementation-independent knowledge in justification structures corresponds to a particular agent's implementation. In the Meta-AQUA system, however,

---

134. This contention concerning planning is also shared by Fox & Leake (1995a).

TMXPs maintain reasoning traces explicitly, and most implementation-dependent knowledge is avoided.

Our approach to using an analysis of reasoning failures to determine what needs to be learned is similar to Mooney and Ourston's (1991) EITHER system, Park and Wilkins' (1990) MINERVA program, the CASTLE system of Krulwich (1993; Collins et al., 1993), Fox's (1995; Fox and Leake, 1995a, 1995b) ROBBIE path planning system, and Stroulia's (1994; Stroulia & Goel, 1995) Autognostic system, but with some important differences. We have focused on the use of meta-models for explicit representation of domain knowledge and of reasoning processes in an integrated, multistrategy reasoning and learning system. Unlike both Mooney and Ourston and Park and Wilkins, we have not assumed a single reasoning paradigm (logic-based deduction and rule-based expert systems, respectively) in terms of which failure situations and learning strategies are characterized. Rather, it is the architecture that provides a basis for a higher-level characterization, which in turn could be implemented in different ways depending on the reasoning paradigm. In fact, Meta-AQUA uses multiple reasoning methods, primarily case-based reasoning.

Birnbaum et al. (1990) focus on the process of blame assignment by backing up through justification structures, but do not emphasize the declarative representation of failure types. They explicitly model, however, the planner. They also explicitly model and reason about the intentions of a planner in order to find and repair the faults that underlie a planning failure (see Freed et al., 1992). Though much is shared between CASTLE and Meta-AQUA in terms of blame assignment (and to a great extent CASTLE is also concerned with deciding what to learn; see Krulwich, 1991), CASTLE does not use failure characterizations to formulate explicit learning goals nor does it construct a learning strategy in a deliberate manner within a multistrategy framework. The only other system to introspectively deliberate about the choice of a learning method is the ISM system of Cheng (1995). ISM optimizes learning behavior dynamically and under reasoning failure or success, but the system chooses the best *single* learning algorithm, rather than composing a strategy from multiple algorithms. ISM does not therefore have to consider algorithm interactions.

Finally, our work focuses on reasoning failures, and not only on performance failures (in both the Collins et al. and Owens' cases, planning failures). Stroulia's approach focuses on a design stance characterization of the reasoner as a device, whereas our approach, as with the approach of Collins, Birnbaum, and their colleagues, takes a more intentional stance toward the reasoner.<sup>135</sup> The analysis of Stroulia (1994), characterizing the ways in which such a device could fail, yields a taxonomy of failure types similar to ours. However,

---

135. Interestingly, Collins et al. (1993) argue from both a design stance and an intentional stance.

like the previous studies, she too does not use declarative characterizations of reasoning failures to formulate explicit learning goals. Furthermore, the CASTLE, Autognotic, ROBBIE, and RAPTER systems are all model-based in their introspective methods, whereas Meta-AQUA is XP-based (case-based). Finally, none of these systems create an explicit plan to learn in the space of changes to the BK. Despite these differences, we emphasize that the approaches enumerated in this section have much in common with IML theory. The following section demonstrates that commonalities also exist within much of the psychological community's research that investigate how persons perceive their own minds, memories and knowledge.

## 11.2 Psychology, Metacognition, and Human Learning

The literature on metacognition (cognition about cognition where the self is a referent) and metamemory (memory and knowledge about one's own memory system and retrieval ability) provides a wide array of influences and support that bear on the research presented here. Our model of introspective learning makes several claims about the nature of learning, reasoning, and introspection that are supported by research in psychology on metacognition. This support includes the emphasis on cognitive self-monitoring, the importance of explicit representation, the emphasis on understanding one's own memory system, and the data demonstrating a person's ability to assess the veracity of their own responses and learning.

### 11.2.1 Cognition and Metacognition

Since Flavell's (1971) coining of the term metacognition, and especially since the seminal work of Flavell and Wellman (1977), many have investigated the phenomenon surrounding cognition about cognition.<sup>136</sup> Of all research on the modern-day concept of metacognition, the child development literature has the longest history (see, for example, Yussen, 1985). Moreover, developmental psychology has reported the most conclusive evidence for the importance of metacognitive strategies and monitoring (see Schneider, 1985; Wellman, 1983). Researchers interested in learning disabilities have studied the metacognitive components of such pathologies. For example, *Part II: Macrolevel Cognitive Aspects of Learning Disabilities* (Ceci, 1987) contains a number of papers relevant to this type of investigation. Research examining the relationship between metacognitive skills and educational instruction have made significant progress. For example, Forrest-Pressley, MacKinnon, and Waller (1985) and Garner (1987) report successful instruction procedures

---

136. Brown (1987) notes that the relationship between text comprehension and metacognitive activities has been studied since the turn of the century, but under the guise of other technical terms.

related to both problem solving and reading comprehension (see also Ram & Leake, 1995, for a related discussion). Most of these works concentrate on applications relevant to teaching in general school environments, although some address specific instruction of the learning disabled. Finally, the social psychology and philosophical communities have all taken considerable interest in individuals' beliefs about their own beliefs and beliefs about others' beliefs (e.g., Antaki & Lewis, 1986; Pollock, 1989a, 1989b).<sup>137</sup>

Wellman (1983, 1985) views human metacognition not as a unitary phenomenon, but rather as a multifaceted theory of mind. Metacognition involves several separate but related cognitive processes and knowledge structures that share as a common theme the self as referent. Such a theory of mind emerges from an awareness of the differences between internal and external worlds, that is, from the perception that there exist both mental states and events that are quite discriminable from external states and events. This theory encompasses a number of knowledge classes considered by Wellman to be psychological variables: *person variables* that deal with the individual and others (for example, cognitive psychologists can recall many facts about cognition, whereas most people cannot), *task variables*, which concern the type of mental activity (for example, it is more difficult to remember nonsense words than familiar words), and *strategy variables* that relate to alternative approaches to a mental task (e.g., to remember a list it helps to rehearse). Finally, Wellman's theory includes a self-monitoring component, whereby people evaluate their levels of comprehension and mental performance with respect to the theory and the norms the theory predicts.

### 11.2.2 Problem Solving and Metacognition

Problem solving is one area where a natural fit exists to studies of higher cognitive processing, such as executive control and monitoring, and where much leverage for metacognitive knowledge could be gained by humans. However, few studies have examined this phenomena explicitly. Some are reported here.

Dörner (1979) reports one of the earliest experiments on the effects of cognitive monitoring on problem solving. The experimental design categorizes subjects into one of two conditions according to how they perform protocols after problem solving. In the introspective condition, subjects reflect out loud about their own reasoning during problem solving, whereas subjects in the control group discuss their solution to the problem in terms of

---

137. Pollock (1989b) distinguishes between knowledge about the facts that one knows and knowledge about one's motivations, beliefs and processes. Introspective multistrategy learning is based on both kinds of metaknowledge; we have argued that introspective access to explicit representations of knowledge and of reasoning processes is essential in making decisions about what and how to learn.

the hypotheses they developed. The experiment itself involves a complicated machine with three lights. Each light can be turned on in four different colors. There are eight push-buttons on the machine with which subjects control the lights and their colorations. The subjects solve ten problems during the experimental trials. Problems consist of an initial state in which the lights of the machine begin operation and a goal state consisting of a different light configuration. Dörner reports that the experimental group performs significantly better than the control group after the third trial. Moreover, Dörner claims that introspective subjects exhibited improved performance during transfer tasks of subsequent experiments, although the details of many of the experiments are lacking.

Derry (1989) offers a comprehensive model of reflective problem solving for mathematical word problems inspired by John Anderson's ACT\* (Anderson, 1983) and PUPS (Anderson & Thompson, 1989) theories of general cognition. Based on such a theory, Derry and her colleagues have developed an instructional system to teach word problems to military servicemen. Prior to the development of this application, Derry performed the following experiment on groups of college students and military personnel. Given an assumption that general problem solving behaviors, such as reasoning from the goal backwards to the solution and means ends analysis, form the bases for human problem solving, the experimenter gathered subject protocols during solution of mathematical word problems. The protocols were classified into 27 categories falling into four basic phases of problem solving: clarifying a problem, developing a strategy, executing a strategy, and monitoring/checking performance. The surprising result was that neither group performed problem solving in a linear fashion, and that most protocols were classified into clarifying and execution phases. The strategy-development and monitoring/checking phases lacked significant protocols.

Delclos and Harrington (1991) report that both subject conditions with general problem-solving skill training and those with problem-solving coupled with metacognitive skill training demonstrate equal performance on a problem solving task. With greater task complexity, though, subjects with the problem-solving/metacognitive training perform better than either a control group or the problem solving training alone group. Also, Swanson (1990) claims to have established the independence of general problem aptitude from metacognitive ability. Subjects with relatively low aptitude, but high metacognitive ability, often use metacognitive skills to compensate for low ability so that their performance is equivalent to high aptitude subjects.

Kluwe (1987) examines the effect of problem-solving task demands on regulatory behavior in subjects aged four through seven. By varying reversibility and irreversibility conditions in multiple puzzle-solving tasks (i.e., the first condition allows pieces of the puzzle to be placed and then moved to alternative locations, whereas the second condition allows no movement once a piece is placed), Kluwe sought to measure the differences in problem solving strategies. Results show that although some activities change regardless



of age (for instance, all subjects increase the duration and amount of problem-solving operations under the irreversibility condition), other activities (such as grouping the pieces) are present in only the older subjects.

Finally, Davidson, Deuser, and Sternberg (1994) present results from a series of studies that show the use of metacognitive abilities correlate with standard measures of intelligence. In their experiments on insight problem-solving they report that, although higher IQ subjects are slower rather than faster on analyzing the problems and applying their insights (not surprising if more processing is being performed), their performance is higher. They argue that the difference in performance is due to effective use of metacognitive processes of problem identification, representation, planning how to proceed, and solution evaluation, rather than problem solving abilities *per se*.

This section has illustrated some of the findings that describe how humans introspect about their cognitive performance (processes) when solving problems and how this ability can lead to improved performance. Although the findings are mixed, and no researcher claims that humans are inwardly omniscient, the results support the relevance of IML theory for modeling intelligence and learning. The next section examines the research into people's ability to understand their own memory systems (content).

### 11.2.3 Metamemory

A large bulk of the research into metacognition pertains predominantly to metamemory knowledge and monitoring of memory performance. Kausler (1991) groups this research into three broad categories: off-line memory self-evaluation, on-line memory self-evaluation, and memory performance monitoring. *Off-line evaluation* of memory concerns a subject's perception of the efficiency and general operation of the subject's memory functions. This is often determined by the use of a questionnaire and then correlated with subsequent memory performance in experiments. For a thorough review of this line of research, see Hultsch, Hertzog, Dixon, & Davidson (1988).

*On-line evaluation* reports a subject's judgement of their performance in a particular memory task. Both feelings-of-knowing (FOK, i.e., judgements of being able to recognize items that are not recalled) and judgements-of-learning (JOL, i.e., judgements while training as to the likelihood of future recall) responses are examples of on-line evaluations. For instance, Lovelace and Marsh (1985) demonstrate that during study, older subjects' judgements of their future ability to perform a paired-associate matching task is less accurate than younger subjects' estimates.

Finally, *memory performance monitoring* is the ability of a subject to associate certain memory strategies with various memory demands or tasks. For example, experiments may

test subjects ability to choose appropriate strategies for memory problems by giving the subject unlimited time to study test words, then measure the amount of time spent in rehearsal. The length of rehearsal time is an index into the subject's knowledge of the behavior necessary to learn the stimulus. Other experiments in this category (e.g., Brigham & Pressley, 1988) measure this ability more directly. Brigham and Pressley report that after practice and feedback, older subjects are less able to determine that a keyword mnemonic strategy is superior to a strategy that generates semantic contexts for recalling word lists than are younger subjects, and therefore do not develop a preference for the better strategy when studying.

Lovelace (1990) subdivides the on-line memory self-evaluation research category into two additional groups: Pre-performance estimates of memory and memory monitoring (not to be confused with what Kausler calls memory performance monitoring). The *pre-performance estimates* paradigm requires subjects to predict subsequent memory performance, and then compares estimates to actual behavior. *Memory monitoring*, on the other hand, concern a subject's ability to evaluate and act upon current memory-states during task performance. These abilities include other subdivisions according to Lovelace: FOK or tip-of-the-tongue phenomena, correctness of response (postdictions), and reality monitoring.<sup>138</sup> FOK judgements correspond to subjects' estimates about their future recognition of currently unrecalled memory items; whereas postdictions concern a subject's belief in the veracity of their responses immediately after they have been given. Reality monitoring is the differentiation between acts performed in the world and those performed in the head (in plans, dreams, imagination, etc.).

Nelson and Narens (1990/1992) present a general information-processing framework for integrating and better understanding metacognition and metamemory. Behind the framework lie three basic principles: 1. Cognitive processes are split into an object-level and a meta-level; 2. The meta-level contains a dynamic model of the object-level; and 3. A flow of information from the object-level to the meta-level is considered monitoring, whereas information flowing from the meta-level to the object-level is considered control. Although the framework is similar to Self's model (see Section 11.1.2 page 268), it differs in that it directly integrates much of the research surveyed in sections 11.2.2 and 11.2.3. The theory addresses knowledge acquisition, retention, and retrieval in both monitoring and control directions of information flow. Monitoring processes include ease-of-learning judgements, JOLs, FOKs and confidence in retrieved answers. Control processes include

---

138. I have used some license in interpreting Lovelace's subcategories to assure consistency with Kausler. Lovelace actually places postdictions in the memory-monitoring subcategory. He considers the pre-performance estimates category to refer to particular tasks, whereas the category that Kausler calls on-line memory self-evaluation Lovelace calls consequences of variation in task or processing and restricts it to metacognitions for how memory works in general.

selection of the kind of processes, allocation of study time, termination of study, selection of memory search strategy, and termination of search. Both acquisition and retrieval of memory items have computationally explicit decompositions in their paper. Although the framework is directed at memory related performance rather than inference-based problem-solving, the distinctions between monitoring and control and the information processing perspective is highly compatible with the views presented in IML theory. We provide a monitoring capacity for detecting failure and a control capability for guiding learning.

#### 11.2.4 Relation of Psychological Research to IML Theory

The preceding subsections examined some of the related research in the metacognition and metamemory communities. This section attempts to reorganize these results with respect to the theory of learning presented in this dissertation. Research regarding metacognition and metamemory processes in humans is associated with our work on introspective learning in at least four specific ways.

First, and foremost, is the emphasis on cognitive self-monitoring. This behavior is the human ability to read one's own mental states during cognitive processing (Flavell & Wellman, 1977; Nelson & Narens, 1990/1992; Wellman, 1983, 1985). Thus, there is a moment-by-moment insight into the content of one's mind resulting in an internal feedback for the cognition being performed and a judgement of progress (or lack thereof). Garner (1987) has argued that metacognition and comprehension monitoring are important factors in the understanding of written text. Reading comprehension is therefore considered to be chiefly an interaction between a reader's expectations and the textual information.<sup>139</sup> Psychological studies have also confirmed a positive correlation between metamemory and memory performance in cognitive monitoring situations (Schneider, 1985; Wellman, 1983). This evidence, along with results from the studies above linking problem-solving performance with metacognitive abilities, directly supports the conviction that there must be a second-order introspective process that reflects to some degree on the performance element in an intelligent system, especially a learning system involved in understanding tasks such as story understanding.

---

139. A special relation exists between metacognition, question asking and text understanding (see Gavelek & Raphael, 1985; Pressley & Forrest-Pressley, 1985). In effect, human learners use question-asking and question-answering strategies to provide an index into their level of comprehension of a given piece of text. This metacognitive feedback helps readers find areas where their understanding of the story is deficient, and thus where greater processing is necessary. Such a perspective supports our ancillary claim that question generation is a key activity in text comprehension and also that meta-level processing is important in such a learning context. As a final tangent, not only is metacognition important in language understanding, it is also important in language generation (i.e., in metalinguistic development; see Gombert, 1992).

Second, our IML theory places a heavy emphasis on explicit representation. Trains of thought, as well as the products of thought, are represented as metaknowledge structures, and computation is not simply the calculated results from implicit side-effects of processing. This emphasis is echoed in Chi's (1987) argument, that to understand knowledge organization and to examine research issues there must be some representational framework. Although diverging from the framework suggested by Chi, IML theory provides a robust form with which to represent knowledge about knowledge and knowledge about process. For example, Section 4.3 illustrated that Meta-XPs can represent the difference between remembering and forgetting (see also Cox, 1994b; Cox & Ram, 1992a), and Section 8.5.4 showed that the Meta-AQUA system can use such representations to reorganize memory indexes when forgetting. In general, forgetting is a neglected issue in AI and computational learning research, yet forgetting is a significant issue in the metamemory literature (Spear, 1978; Wellman & Johnson, 1979). The meta-explanations in our approach are similar to self-explanations (Chi & VanLehn, 1991; Pirolli & Bielaczyc, 1989; Pirolli & Recker, 1994; VanLehn, Jones & Chi, 1992). This research shows that formulation of self-explanations while understanding input examples significantly correlates with subjects' ability to learn from the examples. One difference between the two approaches, however, is that self-explanations are self-generated explanations about the world, whereas meta-explanations are explanations about the self. Despite the differences, experimental results in the psychological literature support the claim that representational structure is important in learning.

Third, because the approach taken by the introspective learning paradigm clearly addresses the issues of memory organization, it can assign blame to errors that occur from mis-indexed knowledge structures and poorly organized memory. As Section 3.3 argues, the memory organization of suspended goals, background knowledge, and reasoning strategies are as important in determining the cause of a reasoning failure as are the goals, propositions and strategies themselves. Thus, memory retrieval and encoding issues are relevant in deciding what to learn and which learning strategy is appropriate. This claim is supported by the metamemory community's focus on organizational features of memory and their relation to the human ability to know what one knows, even in the face of an unsuccessful memory retrieval.

Finally, both metacognition theory and IML theory address the issue concerning a person's ability to assess the veracity of their own responses. In addition, because a person has a FOK, even when recall is blocked, the agent can make efficient use of search. Search and elaboration is pursued when an item is on the "tip of the tongue" and abandoned when an item is judged unfamiliar. This search heuristic provides efficient control of memory and avoids the combinatorial explosion of inferences (Lachman, Lachman & Thronesbery, 1979; Miner & Reder, 1994). Although people sometimes make spurious and biased inferences when assessing their own memories and reasoning, these inferences nonetheless affect people's decisions and thus are important components when modeling human decision-making. For example, current case-based reasoners retrieve the best case and adapt it to the current problem. It is only after the reasoner generates a complete solution and

judges it to be incorrect that the system will attempt to retrieve another case for adaptation. Adding a metacognitive component would allow the case-based reasoner to dynamically judge the progress toward the goal during initial problem-solving (using metacognitive monitoring) and estimate the likelihood of finding another case in memory that applies to the problem (using FOK judgements). Therefore, the reasoner could prematurely decide to “give up” on the initial case and search for a better one by judging the trade-offs involved.

One of the major differences between the manner in which humans learn and the manner in which machines do is that humans perform dynamic metacognitive monitoring or self-evaluation. Humans often know when they are making progress in problem solving, even if they are far from a solution, and they know when they have sufficiently learned something with respect to some goal (Weinert, 1987). They know how to allocate mental resources and, for example, can judge when learning is over. Many reviews (e.g., Chi, 1987; Davidson et al., 1994; Miner & Reder, 1994; Nelson & Dunlosky, 1991; Schneider, 1985; Wellman, 1983) cite evidence for such claims. Research in IML theory is a step in the direction of fully incorporating this metacognitive monitoring capability into artificially intelligent systems.

It should be noted that the learning strategies represented in Meta-AQUA, are at a finer level of granularity than those examined by much of psychology. For example, it would be misleading to assert that the types of learning strategies studied by the metacognition community are similar to index learning, explanation-based generalization, and other learning strategies used in Meta-AQUA. Instead, metacognition research focusses on a person's choice of strategic behaviors at the level of cue elaboration, category grouping, and target rehearsal (in memory tasks); re-reading of text, question generation, and keyword search (in text interpretation tasks); or solution checking, saving intermediate results in an external representation, and comprehension monitoring (in problem-solving tasks). However, many of the results from research on metacognition do support the overall approach we have taken, that of using introspection to support the selection of appropriate strategies in different situations.

Not only is there a relation from psychology bearing on IML theory, but the reverse argument can be made as well. Much of the metaknowledge research in artificial intelligence has focused on knowledge about knowledge and beliefs, or knowledge about the facts that one does or does not know. Much of the metacognition research in psychology has also focussed on similar issues, focussing on cognitive processes, strategies and knowledge having the self as referent. Of particular interest is the psychological research on metamemory, which, in addition to knowledge about knowledge, includes knowledge about memory in general and about the peculiarities of one's own memory abilities. The empirical results obtained from the Meta-AQUA system support the claim that metaknowledge should also include knowledge about reasoning and learning strategies. Experimental results in the metacognition literature suggests that introspective reasoning can facilitate

reasoning and learning (e.g., the studies mentioned earlier: Davidson et al., 1994; Delclos & Harrington, 1991; Nelson & Dunlosky, 1991; and Swanson, 1990). Our research extends these results by specifying computational mechanisms for metacognitive processing, focussing in particular on the selection and use of learning strategies.

## 11.3 Summary and Discussion

This chapter examined some of the research related to IML theory (but not discussed in previous chapters), both from the artificial intelligence perspective and from the cognitive psychology point of view. We described the genesis of interest in computational theories of introspection during the formative years of AI. The logic community has a large part to play in this early research because they established a formalism (and a legitimacy) for the representation of mental states and belief, including beliefs about a system's own beliefs. We also examined the research of the expert system community and others that also claim to be developing introspective systems, but take a different approach. Finally we looked at systems that combine introspective theories with theories of learning. Subsequently, this chapter examined psychological research into metacognition, problems solving, metamemory, and the interactions between each. Both the material on AI theories and that dealing with psychological theories evaluated the relationship between the related research and IML theory and how such research from both fields support the claims presented in this thesis.

Although this chapter represents a relatively broad and cursory examination of the literature and issues, it nonetheless supports the need for further research into the relationship between metacognitive activities, intelligent performance, and learning, especially since there is a natural affinity between high-level cognition and high-level performance in humans. The wealth of ideas in both fields that relate to the issues raised here also argue for a stronger bond between those who build qualitative and mathematical theories of human behavior and those who develop computational models. Yet, before continuing beyond the modest start contained in this thesis, the psychological and AI communities can benefit from better methodological and theoretical foundations. Often, authors use different terms for the same concept (e.g., introspection and reflection),<sup>140</sup> and sometimes the same terms are used in different ways (e.g., metacognition is a multiple overloaded term). Indeed, Brown (1987) has described research into metacognition as a “many-headed monster of obscure parentage.” The description applies equally as well to the many AI approaches that deal with introspection, learning, and the relation between the two.

---

140. But although we have used the terms as synonyms until now, Section 13.2 will make a small distinction between introspection and reflection (see Figure 102).

We have attempted to outline and organize some of the bewildering research as it exists in relation to IML theory and the Meta-AQUA implementation. But until a larger foundation is developed that integrates and supports the many technical approaches and cognitive science perspectives, there may be only limited progress in understanding the computational role of both metacognition in human learning and introspection in machine learning. After the next chapter summarizes the cornerstones IML theory provides, we speculate in an extended epilogue as to some of the forms such a foundation may assume.





## CHAPTER XII

### CONCLUSIONS

*I know  
that I know  
and that I don't know.  
But I forget.*

*I see that I am blind  
and I see the blinding light  
in everything,  
but I forget.*

*I see what I know  
I think  
I know what I see.  
But sometimes  
I forget.*

*And This is the Way  
It should be.  
At the end of every forget  
I remember.*

—Lonny Brown (cited in Ram Das, 1971), p. 98.

Many dissertations are written from an extremely narrow and focussed perspective; this one has attempted to start with a narrow computational problem, but then to examine the problem from a broader cognitive science perspective. We examined topics ranging across a wide spectrum, from representations to algorithms and from historical context to implementation. The attempt to be general in our theory of introspection has dictated that the research at least consider comprehension, problem solving, and learning in some depth. We take up again these three processes in an extended epilogue to come, but first, we review what this work has accomplished so far. The goal of this chapter is to briefly contextualize the contributions and major points by which the reader might remember this research.

The thesis contains four major parts. Part One introduced and motivated the problem of constructing a learning strategy. Within it Chapter I defined the main problem and outlined its solution. This chapter introduced the issues addressed by all subsequent material. Chapter II then explained the distinction between a process theory and a content theory of cognition. Following these preliminaries, Part Two developed a content theory of learning and introspection (Chapter III gave the content and Chapter IV provided the representation for the content); whereas, Part Three detailed the corresponding process theory (Chapter V set it up theoretically and then Chapters VI and VII provided an extended example that

made the process theory concrete). Part Four finally inventoried the implementation of the theory (Chapter VIII) and evaluated the theory using the implementation (Chapter IX). Part Four concludes with future (Chapter X) and related (Chapter XI) research and a final closing (Chapter XII).

Section 12.1 explicitly lists the main points of this thesis. The subsequent four sections (12.2-12.5), will expand on each of the four main parts to the dissertation, briefly reviewing the main contributions and putting them into context for the reader. Section 12.6 finishes by reiterating the major contributions of the thesis.

## 12.1 Major Points of the Dissertation

To be as explicit as possible, we enumerate our most important claims here and follow with a brief summary of the points in context of an outline of the dissertation (within the summary the points are marked likewise and printed in italics). If the casual reader takes from this dissertation nothing but a litany of slogans, then the following list represents the thesis in convenient sound-bite form, ordered by decreasing importance. I challenge the reader to consider them at leisure, however, and if interested, to search the appropriate chapter number (in parentheses) for the source of these claims in order to independently determine their reasonableness.

1. The question of learning-strategy construction is important because if ignored, a system that conjunctively calls learning algorithms can incur negative interactions. (I, VII, IX)
2. Deciding what to learn is a necessary process, if learning is to remain effective, that is, if negative interactions between learning methods are to be avoided. (VI, IX)
3. Explaining reasoning failure (blame-assignment) is like diagnostic troubleshooting: case-based introspection is a symptom-to-fault mapping from failure type to failure cause. (V, VI)
4. Learning is like nonlinear planning: The task is to create a plan of learning steps that achieve desired changes to the background knowledge. (VII)
5. To learn effectively requires introspection and reasoning about reasoning. (I, V, IX)
6. To reason about reasoning effectively, the reasoning failure must be represented

declaratively in explicit Meta-XP structures. (IV)

7. Failure types should be derived from a model of the reasoning process rather than by generalization from the researcher's intuitive list of all likely failures. (III)
8. In a computational theory of introspective learning, a special relation exists between the content and process theories because the content theory must represent what the process theory describes (i.e., process failures). (II)
9. We are starting to give machines an ability to think about themselves despite what the critics (e.g., Searle, 1992) think. (I-XII)

## 12.2 Part One: Motivations and Defining the Problem

Chapter I set up the basic arguments used by the thesis in establishing a theory of introspective multistrategy learning. It provided a motivation for why an introspective approach to learning is effective, especially when integrating disparate learning methods. It then narrowed the thesis focus to the learning-strategy construction problem; that is, it posed the question “How can an intelligent system select and order learning methods given a performance failure?” ①*The question of learning-strategy construction is important because if ignored, a system that conjunctively calls learning algorithms can incur negative interactions.*

The chapter placed this question in a machine learning context and asked what significance it has for models of human learning. The chapter outlined a solution to the strategy construction problem and argued that to answer the question a number of interrelated subquestions must be answered first. Figure 100 lists brief answers for each of the subquestions in the order they were entertained by this document.<sup>141</sup> The questions represent the major research problems considered and the answers represent the specific contributions.

Given the introduction to IML theory and the Meta-AQUA implementation in Chapter I, Chapter II provided two examples that provided tangible material to illustrate many of the abstract arguments put forward in subsequent chapters. We then explained the differences between content and process theories and related the distinction to the performance domain of story understanding. A content theory specifies the salient features for representing knowledge in a domain; whereas, a process theory specifies the transformations on

---

141. See Figure 9 on page 14 for the goal structure of the items in this list.

- **Q6:** What kinds of reasoning failure exist?  
**Ans:** Contradiction, impasse, false expectation, surprise, and unexpected success.
  - **Q4:** What can cause reasoning failure?  
**Ans:** Knowledge, processes and goals or the way they are indexed in memory; Environment or the way to which it is attended.
  - **Q7:** At what level of granularity should reasoning be represented?  
**Ans:** At enough detail to support learning from failure.
  - **Q5:** How to represent mental states and reasoning mechanisms?  
**Ans:** Use meta-explanation patterns (Meta-XPs)
  - **Q3:** How to explain a reasoning failure?  
**Ans:** Case-based introspection: Maintain trace of the reasoning; when failure occurs, retrieve past case of meta-reasoning (Meta-XPs) about the class of failure; apply to trace.
  - **Q2:** How to automate learning decisions?  
**Ans:** Use bound Meta-XP (above) to post explicit learning goals.
  - **Q1:** How to choose or construct a learning strategy?  
**Ans:** Treat strategy selection as a planning problem: Achieve learning goals (above) with nonlinear planner and learning algorithms encapsulated as STRIPS-like operators.
- 
- **Q0:** How can the research be evaluated?  
**Ans: A.** Meta-AQUA implementation - Covered LISP learning protocol; hand-coded examples; Elvis World empirical study.  
**Ans: B.** Meta-TS implementation - Covered human data from troubleshooting circuit boards.

---

Figure 100. Research goals and results (contributions)

that knowledge. ⑧*In a computational theory of introspective learning, a special relation exists between the content and process theories because the content theory must represent what the process theory describes (i.e., process failures).*

## 12.3 Part Two: Content Theory of Introspective Multistrategy Learning

Four of the eight research questions listed in Figure 100 address issues of representation and content. Chapter III tackled two of them while Chapter IV handled the remainder. The purpose of Chapter III was to describe what needs to be represented (i.e., the content). Because the task of learning from failure involves mapping from failure symptom to failure cause, the representational focus is not on the content of some task domain in the world, rather the theory must represent the second-order learning domain of reasoning failure. Thus, this chapter provided a simplified but general model of reasoning from which failure can be analyzed. The question of what kinds of failure exist (Q6) can then be answered by exhaustively deriving the implications of the model.

⑦*Failure types should be derived from a model of the reasoning process rather than by generalization from the researcher's intuitive list of all likely failures.* Our analysis of the reasoning model showed that failure consists of contradictions, impasses, false expectations, surprises, and unexpected successes. Given these failure symptoms, the remainder of the chapter analyzed the kinds reasoning faults that can cause such symptoms. We argued that knowledge, goals, processes and the environment all contribute to failure. For each of these categories, we additionally showed that each can have a selection or organizational component. This then determines a matrix of causal factors that explain reasoning failure (Q4).

Chapter IV provided specific knowledge structures that declaratively represent the content of reasoning failures described in the previous chapter. To represent failures, the details of mental events and mental states are captured just as standard AI knowledge representations capture the details of physical events and states. The most important details are the causal relationship between chains of reasoning, and thus not all details of reasoning need be represented. Rather, the granularity of representation must be fine enough to support explanations of failure and the learning process (Q7).

⑥*To reason about reasoning effectively, the reasoning failure must be represented declaratively.* To discuss how to declaratively represent the mental domain, we first examined forgetting. We argued that both the logic and CD formalisms are insufficient when representing such reasoning failures because they do not express well the causal nuances involved (e.g., the difference between forgetting due to missing knowledge and that due to mis-indexed knowledge). We then described how IML theory represents an entire class of

retrieval failures with a single meta-explanation pattern (Q5). The Meta-XP knowledge structure comes in two varieties; TMXPs represent how reasoning fails and IMXPs represent why reasoning fails. After providing a basic representational vocabulary, the chapter described how to represent reasoning success and all symptoms of reasoning failure. These representations compose the types of failure that a system must be able to detect and about which it must reason (i.e., contradiction, impasse, false expectation, surprise, and unexpected success). Representing them explicitly simplifies the reasoning and learning tasks.

## 12.4 Part Three: Process Theory of Introspective Multistrategy Learning

Chapter V began by reviewing the theoretical assumptions of IML theory. It placed the theory in the context of the multistrategy learning framework and developed a process model of both understanding and learning. This process theory describes how a system can use the representations described in Part Two when learning from failure. Learning is considered a four phase procedure: blame assignment, deciding what to learn, learning-strategy construction, and learning-strategy execution. The first two phases represent a case-based approach whereas the latter two consist of non-linear planning techniques. A functional justification of this learning model holds that ⑤*to learn effectively requires introspection and reasoning about reasoning.*

Using the initial example from Chapter II, the next two chapters worked through the learning process that allows a system to create a learning plan in response to a performance failure. The performance task is story understanding. When Meta-AQUA incorrectly explains an anomaly in any story it reads (or makes other errors), the system must explain the error. ③*Explaining reasoning failure (blame-assignment) is like diagnostic troubleshooting: case-based introspection is a symptom-to-fault mapping from failure type to failure cause* (Q3). The system examines the prior reasoning that preceded the failure (represented in a TMXP), retrieves an introspective explanation of the failure (represented in an IMXP), and applies this case to the current failure to form a causal graph of the points most likely to be responsible for the failure. With this explanation, the system can then generate a set of learning goals that, if achieved, will correct the flaws in the system's BK that are responsible for the failure (Q2).

④*Learning is like nonlinear planning: The task is to create a plan of learning steps that achieve desired changes to the background knowledge.* Chapter VII looked at how a non-linear planner can create a learning plan in response to the learning goals spawned by the system and as discussed by Chapter VI. To examine the relevance of the planning metaphor to learning, we showed that Sussman's anomaly has a correspondence when planning to make changes in a system's background knowledge and thus interactions exist. Like goals to achieve On (Block-A, Block-B), a system can plan to achieve goals such as Differentiate (Expected-Explanation, Actual-Explanation), but it must be careful that plan

steps do not interact. A learning plan consists of a partially ordered sequence of steps, where the primitive steps in the plan represent calls to specific learning algorithms. The creation of a non-linear learning plan represents an answer to the question of how to construct a learning strategy in a multistrategy learning context (Q1). Once assembled, the plan is simply executed by calling the appropriate learning methods.

## 12.5 Part Four: Evaluation and Implementation of the Theory

The description of the example by the previous two chapters was at a somewhat high level. Chapter VIII provided more of the implementational details that explicate the workings of the Meta-AQUA system. It included separate subsections on the performance, input, memory, and learning subsystems. Most importantly, the section on input described the Tale-Spin story generator that was used to generate Elvis World stories. These stories were used in the subsequent chapter on evaluation.

Chapter IX posed and examined two specific hypotheses in order to answer the question of how to evaluate the theory presented by the dissertation (Q0). Hypothesis One is that introspection facilitates learning. In an extensive empirical study, Meta-AQUA performed significantly better in a fully introspective mode than in a reflexive mode in which learning goals were ablated. A novel performance metric used partial credit for establishing the amount of understanding the system exhibited when explaining anomalous or otherwise interesting input. In particular, the results lead to the conclusion that *deciding what to learn is a necessary process, if learning is to remain effective, that is, if negative interactions between learning methods are to be avoided.*

Hypothesis Two claims that IML theory represents a sufficient model of human learning. We showed the applicability of IML theory to human learning by modeling two real-world tasks to which the theory applies (programming in LISP and electronics troubleshooting). The minimal modifications necessary to get Meta-AQUA to cover the LISP learning protocol fragment suggests that IML theory is a sufficient model of introspection. The alternate implementation of the Meta-TS system showed the generality of the theory as a human model. The results presented in this chapter also support the hypothesis that the failure symptom taxonomy as described by Chapter III is a reasonable categorization for both artificial and natural reasoners because these failure types are instrumental in the Meta-AQUA and Meta-TS systems from which the results were reported. *We are starting to give machines an ability to think about themselves despite what the critics (e.g., Searle, 1992) think.*

## 12.6 Contributions

Figure 100 enumerates several contributions that stem from this research. As stated in the thesis introduction, the narrow research goal was to establish a specific solution to the learning-strategy construction problem (i.e., given a performance failure, assemble a calling sequence of learning methods while avoiding negative interactions). Our major contribution was to develop a means for constructing a learning strategy by treating the learning task as a planning problem. To perform this we showed how to encapsulate learning algorithms as operator schemas in Tate's (1976) Task Formalism and developed a taxonomy of learning goals that direct such operators. We also made contributions to the blame assignment problem that enable machines to formulate learning goals. This entailed the specification of a case-based method of introspection wherein failures could be explained by mapping from symptoms of failure to the underlying causes. Supporting this method we developed the meta-explanation pattern representational formalism, a taxonomy of failure causes, and a classification of failure symptoms. In its entirety, this thesis presents a content theory and process theory of both question-driven story understanding and introspective multistrategy learning.

A more general contribution was to begin to establish empirically the conditions under which such introspective processes are productive during reasoning and learning. We demonstrated that in the presence of learning algorithm interactions, non-introspective reasoning ran the risk of poor performance. Thus, when such negative interactions are present, it is imperative that methods such as the ones presented here be used to avoid these problems.

The contributions of this thesis are extremely important if AI is to create systems that can scale their performance in the real world. Rather than attempt to develop generalized "weak" method of learning, this thesis lays the groundwork for technology that can integrate the multitude of "strong" methods currently available in the machine learning community. Especially when complex intelligent systems are drafted to perform increasingly complex tasks in dynamically changing environments, performance failures are inevitable. The advantages of a system that can reason about these failures and, as a result, automatically assemble a sound strategy to deal with these misfortunes cannot be overestimated.

Here we have also taken a first step toward providing machines with an integral component of intelligence: the sense of self. This faculty is clearly one of the most unique and important of those human qualities that separate us from the rest of the natural world.<sup>142</sup> But to make significant progress beyond the modest beginnings presented here requires an

---

142. If this assertion is not self-evident, then see Metcalfe and Shimamura's (1994) preface to *Metacognition: Knowing about knowing*.



integration of the many aspects of intelligence that we have only partially addressed. Introspection and learning are not isolated atomic processes and do not make computations on a singularly represented brand of knowledge. Instead, they represent amalgams of processes and knowledge that together give humans an insight into our strengths and weaknesses and that provide us with leverage when dealing with the world and its many challenges. The epilogue provides some optional words in speculation as to the possible nature of a more full integration of these components for intelligent learning machines.



## CHAPTER XIII

### EPILOGUE

*I wish a robot would get elected president. That way, when he came to town, we could all take a shot at him and not feel too bad.*

—Deep Thoughts  
by Jack Handey.<sup>143</sup>

But would the above assassin not feel different if she knew that the president-robot had a sense of self? Like the computer who developed a personality in *The moon is a harsh mistress* (Heinlein, 1966) and then reverted to its original state at the novel's close, wouldn't we all feel badly if we had first known the robot through conversations and interactions? During conversations, the robot might explain its goals, biases, past experience, and perhaps an opinion of itself. The robot might even have the goal of self-preservation.<sup>144</sup> What if, as envisioned by McCarthy, it really had consciousness?<sup>145</sup> When I read Heinlein's story, I sympathized with the machine. If the robot-president was real, an assassination might be quite disturbing. Although the need to answer the above questions is far from immediate, research into the mental lives of machines and people is starting to blur the line between science fact and science fiction.

The aim of this chapter is to secure the dissertation in the larger context of creating an integrated architecture of intelligence and learning. I make some speculative comments with respect to this problem by examining the kinds of processes (Section 13.1) and the kinds of knowledge (Section 13.2) that must be integrated for a general thinking machine.

---

143. Handey (1992).

144. Weld and Etzioni (1994) are working on formalizing the first of Asimov's (1942) Three Laws of Robotics. The first law directs robots to avoid harm to humans, the third is self-preservation.

145. Note that this work does not pretend to constitute a computational theory of consciousness, although it is related. Others have much to say on the subject, however (e.g., Chalmers, in press; Dennett, 1992; Edelman, 1992; Hofstadter, 1979/1989; Rosenfield, 1992).

Obviously such proposals are monumental, and therefore success cannot realistically be defined in terms of a final product. Rather, it is theoretical *progress* towards Turing's vision of a program that can pass some variation of an intelligence test that, although provocative, is not an unreasonable goal (Section 13.3).

## 13.1 The Processes:

### Integration of problem solving, understanding, and learning

This thesis has examined an introspective approach to learning within the context of a story understanding task. We have also experimented with using the same approach to learning when problem solving given two troubleshooting tasks. The Meta-AQUA system was modified to model learners in a LISP programming task and the Meta-TS system modeled human protocols while learning an electronics diagnostic task. Chapter V briefly discussed the relationship between problem solving, understanding, and learning from a structural perspective. That is, the section compared and contrasted features of each process and drew parallels between such features. However, the chapter never specifically proclaimed how the processes might be integrated into a gestalt or how they might interact. Moreover, Section 5.4 claimed that the multistrategy paradigm was a natural one for integrating the performance and learning tasks, but it was not very specific either. This section considers again this integration question and speculates as to a possible functional merging of problem solving, understanding and introspective learning.

Both Birnbaum (1986) and Wilensky (1983) discuss integration approaches to problem solving (in the guise of planning) and understanding. Birnbaum argues convincingly that both planning and understanding processes need to use an early integration of bottom up and top-down information to constrain the explosion of inferences inherent in both processes. Wilensky asserts that knowledge of goals and plans is instrumental in the successful performance of both planning and understanding tasks. Yet, Wilensky and Birnbaum both discuss planning and understanding as otherwise unrelated tasks. In their writings, they do not attempt to specify an integrated architecture within which both processes could be unified. The two researchers also view the scope of understanding as being limited to natural language (although Wilensky briefly mentions the possibility of understanding based on video tape images). Finally, they are viewing understanding as the task of comprehending other agents or characters in a story, rather than understanding the actions of the self. The major contribution of their work is the recognition that processes such as these must integrate many knowledge sources and that many processes need to use the same knowledge sources. Processes cannot be insulated from each other by the overly-restrictive modularity of computation. This section furthers the integrative view, arguing that understanding and problem-solving are intimately related in the following ways.

Newell and Simon (1972) postulate an initial problem solving stage that translates an

input problem statement into an internal problem representation and problem space.<sup>146</sup> Although Newell and Simon downplay the significance of this stage (p. 850), Greeno and Riley (1987) suggest that it is at this point in the process that much of the potential for solving the problem is either established or, if the wrong understanding of the problem is obtained, is lost. They argue that better problem solvers have a better problem-understanding ability, and it is at the early stages that understanding becomes critically engaged in the problem-solving process. Furthermore, Greeno (1977) likens the understanding of a problem statement to the understanding of a sentence. Both include the construction of a representation that captures the major concepts in either the problem statement or the sentence. The understanding of the problem creates a representation for the solution that includes meaningful relations between the solution steps and relations to parts of the problem statement. Greeno and Riley speculate that this kind of understanding is metacognitive and consists of strategies for performance.

As with understanding a story, the schematic structures used to interpret a problem are similar to scripts (Schank & Abelson, 1977). The development of a child's ability to solve problems is related to the acquisition and refinement of these problem-solving schemas (Greeno & Riley, 1987). This result is compatible with the assertions presented in Section 8.5.3, "Learning about higher-order knowledge," starting on page 201. This section asserted that an important learning task in Meta-AQUA was to refine script and other schematic knowledge structures and was not limited to the development of conceptual categories alone.

This section, however, takes the relationship between problem-solving and understanding a few steps further. The understanding processes involved in problem solving is not limited to making sense of a structured problem statement provided to a learner in a formal setting (e.g., classroom or work environment). In an unconstrained world, a significant task component is to identify those situations that require problem solving. That is, a learner must be able to understand when novel problems exist, must be able to formulate an understanding of the circumstances that can lead to a solution, and must be able to generalize the conditions so that similar problems can be recognized in the future.

As will be remembered from Chapter V, we consider reasoning in general to be a type of generate-and-test procedure with an initial interestingness identification phase. In this framework, problem solving, as well as understanding, can be cast in multistrategy terms. Understanding involves detecting an interesting input and then choosing or constructing a strategy with which to explain the input. Likewise, problem solving requires the reasoner first to determine something interesting that constitutes a problem and then to choose a

---

146. For a few more details about their model, see the discussion that opens Section 5.2 on page 107.

problem-solving algorithm or to assemble a problem-solving strategy. Sometimes, the problem is as simple as not being able to perform some previous task, so a problem goal arises as an impasse (Etzioni et al., 1992; Newell, 1990). At other times, problem-solving goals may arise from role, interpersonal, and life themes (Schank & Abelson, 1977). But, most interestingly from a learning point of view, goals may arise because of the lack of knowledge (Ram, 1991). That is, a problem may be formulated when an item is interesting because one does not know much about a subject.<sup>147</sup> Ideally and as with learning, the reasoner might use some method of introspective deliberation to create its reasoning strategy in all of these cognitive tasks.

Parallels exist throughout the framework in which these two processes are cast (see Figure 101). For both understanding and problem-solving, if no significant or interesting input exists in the environment, the system processes the input in a “mindless” fashion. The understander skims the input, whereas the problem solver acts reactively (i.e., just “goes through the motions” of behavior). Given interesting input, however, the understander must generate an explanation and the problem solver must generate a solution. In both generation stages the reasoner can select a method and assemble a reasoning strategy. Figure 101 shows a few possible example choices. Finally, given an explanation or a solution, the reasoner must verify the outcome of reasoning.

In this last stage, the two processes are mutually related to each other. When trying to understand an input, the understander often poses a hypothesis that requires validation. To validate the hypothesis the understander might choose to devise a test or create a plan to falsify the hypothesis. The creation of such a plan requires problem solving. Conversely, when attempting to plan for a given goal, the problem solver needs to understand whether the actions carried out during plan execution are indeed furthering the pursuit of the goal. A comprehension ability is required to provide feedback to the problem solver during plan execution (again, see Figure 101).

Functionally, reasons exist that encourage such a proposed integration.

- *The function of understanding is to make an input coherent.* When it is not coherent, either an anomaly exists that need to be explained and learned from, or a problem

---

147. For example, an agent may be interested in learning about a friend’s artistic ability because it is a novel, and therefore intriguing, behavior. To acquire such skill requires planning and problem-solving activities. As the agent learns more about a subject, the topic may become less interesting. If it becomes a well-practised or habitual behavior, it may actually become boring. Thus, an agent may not only form goals out of mundane necessities (e.g., career-related goals), but may create new goals given a thirst for knowledge and new experience.

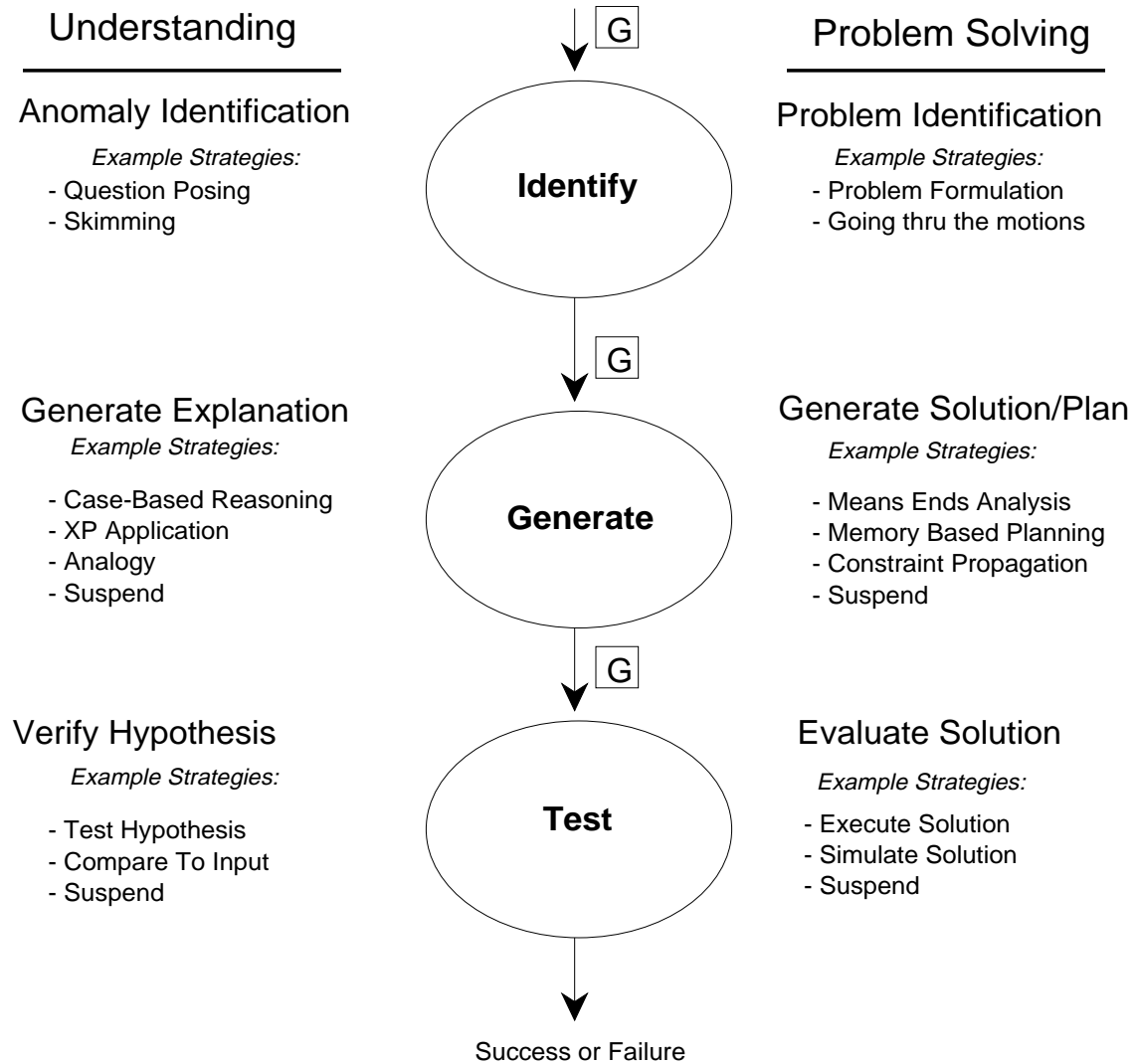


Figure 101. Relationship between understanding and planning

exists that must be solved (i.e., the plan is not going as intended or opportunities arise when going through the motions).

- *The function of problem solving is to establish a desired state.* When the state is the validity of a hypothesis posed by the understander, a plan may be devised by the problem solver to achieve that state of confidence. Impasses encountered during a plan mandates a new learning goal.
- *The function of learning is to improve the performance task* (either understanding or problem-solving). When understanding detects a gap in the knowledge of the reasoner and learning is required, problem solving may be used in the execution of the learning plan.<sup>148</sup>

The processes of learning, problem solving, and understanding mutually benefit and are share relationships to each other. Thus, the study of one process in isolation may leave enough questions unanswered that research results may or may not sufficiently generalize to human behavior or may not scale when engineering intelligent systems that are expected to operate in unconstrained environments.

Much of the research performed by the AI and cognitive science communities over the years has concentrated on problem solving and planning in relatively well-developed paradigms that possess optimal or provably correct solutions. Problem solvers are explicitly given their problems in tangible formats. The reasoner seldom has to consider whether the problem is worth pursuing, or what other problems might be worth tackling. Understanding, on the other hand, has received less attention. Furthermore, much of the research into understanding (e.g., research into story understanding) has treated the comprehension process in isolation without considering either action or the environment, and without considering the supertasks under which the understanding task is situated (i.e., for what purpose do story understanders read?). But research into the alliance of understanding and problem solving comes closer to the locus of intelligence than research into either alone. Surely, any unified theory of learning must eventually deal with learning stemming from both behaviors. Although the comments supplied in this section are highly premature, the suggestions fit well into the framework provided by this research.

---

148. For example, the missing information may be contained in a book, and so a knowledge goal is created to acquire it. The problem solver creates a plan to find a book with such information and to obtain the book. A story understander may then be used to extract the knowledge required by the learning plan in support of the learning goal.



If for no other reason, it is at least an interesting research challenge to begin to consider what the understanding process has to do with problem solving, and vice versa, and moreover, how the integration of each affects learning. Towards this aim, a number of researchers have appealed for general integrated architecture of intelligence and learning (e.g., Newell, 1990; Plaza, Aamodt, Ram, van de Velde, & van Someren, 1992; Pollock, 1989a; Ram, Cox, & Narayanan, 1992; Ram & Jones, 1995; VanLehn, 1991a). In support of this goal, an integration of knowledge sources must be considered as an important sub-goal.

### **13.2 The Knowledge: Integration of world knowledge, metaknowledge and self-knowledge**

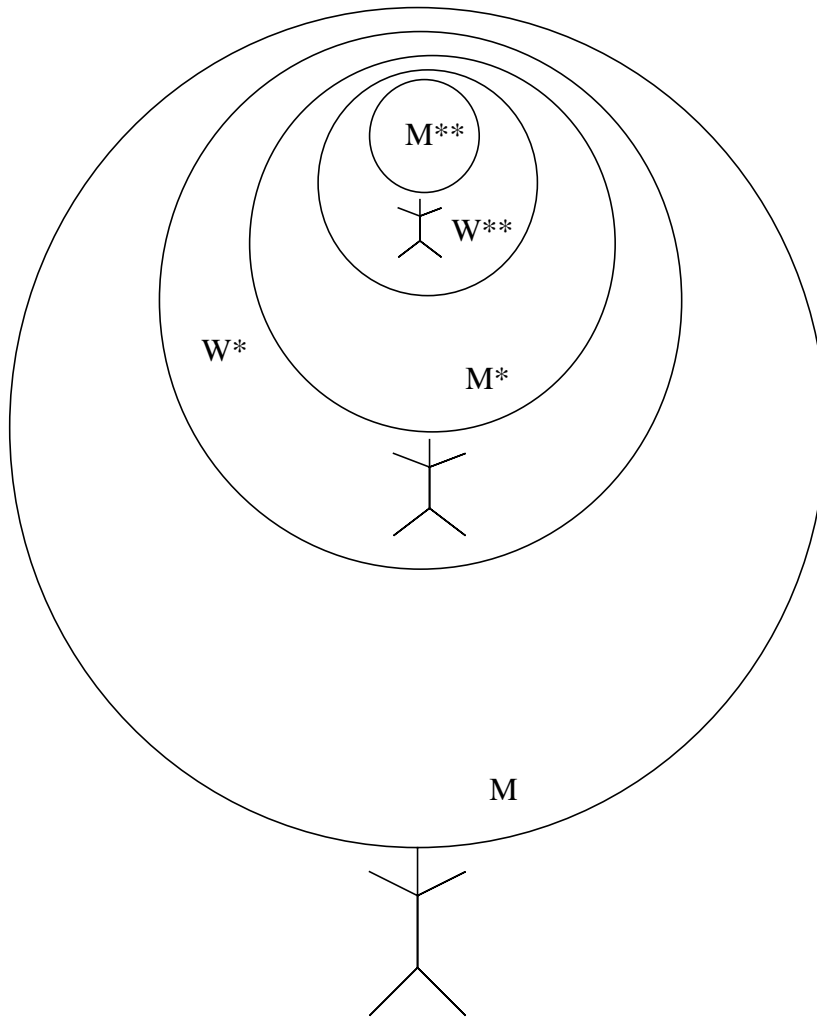
One of the most remarkable papers on ontological matters is a four and one half page desiderata by Minsky (1965, 1968a) concerning the mind-body problem and the assertion that human understanding is essentially the process of executing some model of the world. Minsky's thesis is most interesting because it includes the modeling of not only the world, but the self (the modeler) as well. Thus, there is *W*, the world, and *M*, the modeler<sup>149</sup> who exists in the world. Although Minsky's models are procedural, Figure 102 depicts the knowledge divisions implicit in his scheme. The model of the world is referred to as *W\**. *W\** is used to understand and answer questions about the world. So to answer questions about oneself in the world, it must also be the case that there exists within the model of the world, *W\**, a model of the modeler, termed *M\**. One should conceive of *W\** simply as the agent's knowledge of the world, and likewise, *M\** as the agent's knowledge of itself in the world. Furthermore, as Minsky notes, one must have a model of one's model of the world, or *W\*\**, in order to reason about and answer questions concerning its own world knowledge. Although Minsky does not label it as such, the kind of knowledge embodied in this model is typically referred to as metaknowledge. Finally, *M\*\** represents the agent's knowledge of its self-knowledge and its own behavior, including its own thinking. Within *M\*\** one might include most metacognitive knowledge of person variables (see Wellman's theory of metacognitive variables on page 273), at least concerning the self. It would have a semantic component like "I am good at general memory tasks," as well as episodic components such as knowledge gained through monitoring (e.g., "I just solved a problem by remembering a similar past solution."). Again, although Minsky does not refer to it as such, *M\*\** represents introspective knowledge.

This taxonomy can be extended into a framework that supports introspective learning when viewing not only oneself in the world, but others (*O*) as well. Wellman (1985) claims

---

149. In the language of the times, *M* actually stood for "man."

W



W = World

W\* = World Knowledge

W\*\* = Meta-Knowledge

M = Modeler

M\* = Self (Reflective) Knowledge

M\*\* = Introspective Knowledge

Figure 102. A taxonomy of knowledge  
(Adapted from Cox, 1992).

that children come to develop an understanding of themselves and the workings of their mind by observing other agents in the world. By observing others, children first learn the distinction of internal versus external reality. A naïve theory of mind first emerges from a theory shift from a simple desire-psychology to a belief-desire psychology (Wellman, 1992). This shift occurs when children observe that the actions of others do not always conform to the simpler theory. For example, two agents may have the same desires, yet perform different actions. The simple concept of desire is not sufficient to explain the anomaly. The conflict can be explained, however, if agents have opposing beliefs. If two people are hungry (desire food), they may look for food in different places because one believes the food to be in the cupboard, whereas another believes it to be in the refrigerator. Thus, children develop a theory of mind, mental events, and mental states (beliefs). Therefore, the knowledge framework that we propose should contain a modeling of others in the world as well as the self, hence  $O^*$ . The taxonomy is complete by including  $O^{**}$ , which is knowledge of one's knowledge of and confidence in social interactions.

Additional distinctions exist that must be integrated. One such dimension is the distinction between internal and external versions of each of the taxonomic categories. For example, there is reflective knowledge concerning one's own actions in the world as well as knowledge of one's own thinking. These are both reflective knowledge in  $M^*$ , but it seems reasonable to distinguish the two kinds since one is usually objective, whereas the other is often subjective. Moreover, people can learn things about themselves, both about their own external behavior and their own internal workings of reason, via the knowledge of verbal reports of others (that is, via  $O^*$ ). For example, rather than learning from introspection or self-monitoring, people may gain some insight into their own reasoning ability by listening to lectures in cognitive psychology. It is, an open question, however, as how best to integrate such nuances.

Contrastingly, Chi (1987) develops and utilizes a consistent distinction between process and state, emphasizing the representational difference between processes, such as cognitive strategies, and mental states, particularly the knowledge states of the individual. In her taxonomy, processes are represented procedurally in condition-action rules, whereas knowledge is represented declaratively with semantic, propositional net structures. The analysis further divides process into domain-specific procedures, domain-independent, goal-based strategies, and meta-strategies, which evaluate the applicability of ordinary strategies. Declarative knowledge is likewise subdivided into domain-specific knowledge, general knowledge and metaknowledge, that is, knowledge about other propositions or about the self. Furthermore, her analysis argues that a difference exists between meta-procedures and ordinary procedures; whereas no significant differences exist between declarative knowledge and meta-declarative knowledge. The former requires a function be an argument to another function, so it is second order rule that requires extra execution overhead, while declarative knowledge is uniformly stored as memory nodes in the same manner as normal knowledge, and is therefore retrieved with no additional overhead. These differences result in an alternative interpretation of the metacognition literature, whereby

some effects can be explained simply by presence or lack of knowledge, rather than requiring reference to strategy or metalevel issues at all.

Finally, an additional issue exists as to the role of general versus domain-specific problem-solving knowledge. Derry (1989) considers general problem-solving knowledge to be metacognitive. It is unlikely, however, that this proposition is unequivocally valid. Perhaps it is indeed knowledge about knowledge, but as Chi (1987) suggests, facts that simply happen to be about the self or about other knowledge are much like ordinary knowledge. They are retrieved as any other fact is retrieved, adding no special properties to their content or overhead to their usage. There seems to be a difference between the way precompiled knowledge is obtained and generalized about the self and the way on-line insight of the self is processed. Minsky’s modified taxonomy has nothing to say concerning this orthogonal concern. Therefore, while the taxonomy may help to visualize the divisions of knowledge and to avoid category errors in metacognitive research, it alone is not sufficient when describing all elements that bear on relevant aspects of metacognition. Together with Wellman’s (1983) and Chi’s (1987) taxonomies of metacognitive knowledge, the taxonomy of Figure 102 modified to include O, O\* and O\*\* may help draw the line between what is and is not a factor or component when investigating the role of introspection in human performance and learning.

The following section examines what it means to put all of these concepts together and speculates on some popular, yet serious, research goals. Although it considers some thought provoking questions, the tone to be set is one of suspended judgement. The concluding section of this thesis is the one section that deliberately provides no explicit answers or strong opinions; instead, it requests that the reader decide.

### 13.3 The “Turing Test”

Many papers have discussed Turing’s (1950/1963) famous test of computer intelligence. In his essay, he poses the question “Can machines think?” and subsequently transforms it into a more operational question. Turing’s *Imitation Game* is designed to test whether or not a computer can perform as well at imitating a woman as another man, given a human judge and the cloak of video-terminal output. If by posing questions to both agents the judge cannot distinguish on an above average basis the man’s responses from the program’s responses, then the computer is considered intelligent and capable of thought.

A number of researchers have interpreted this test in various ways, including removing the requirement of gender association. In recent years, Hugh Loebner, President of Crown Industries, established a \$100,000 cash award for anyone who can design a program that passes the Turing test in a limited domain (Loebner, 1994; Zeichick, 1992). Instead of

an open-ended question/answer session between judge and contestants, the programs answer questions in a specific knowledge domain chosen by the programmer. Even granted these concessions, the Loebner Prize remains unclaimed. The programs seem to be getting close enough to winning, however, that Loebner is making more strict demands on the contest rules. Thus for the next five years, the contest will be conducted without domain limitations (Loebner, 1994). Despite Turing’s belief that sufficient programs could be built by the end of the century, research must cross significant hurdles to pass the unconstrained test.

A number of problems exist even with restricted tests. Perhaps the foremost weakness of this test as a measure of intelligence is that it evaluates output alone, with no regard to how the output was obtained. If a performance task’s output constitutes the only measure by which one can conclude that a program possesses intelligence or a capacity for thought, then one must conclude that successful chess-playing programs that win by brute force search of board positions must also possess thought. But certainly human chess champions do not win through search alone. So to improve the Turing Test, the programs should be required to agree with known human limitations and to conform to accepted psychological theories. This makes it less likely that a system can pass the test by simply brute force computation.

Schank (1986, and Searle, 1980, 1990) also notes the weakness of output as an evaluation metric in the Turing Test. Schank claims, however, that language is the only practical method of judging intelligence because judges can open the heads of neither people nor machines. Moreover, he asserts that explanation is the ultimate criterium of intelligence. To improve Turing’s Imitation Game, then, Schank proposes the *Explanation Game*. In this test, an agent’s ability to explain its own reasoning is the focus of evaluation. Both the human and the machine are given mental tasks to perform. The judge then can ask how they actually performed the mental behavior or how they produced the constructed outcomes. Furthermore, answers are not absolutely correct or incorrect in the Explanation Game. For a contestant to explain *what* it does is passing the test at the level of only *making sense*. To explain *why* it reasons as it does is to pass at the level of *cognitive understanding*. The upshot of the test is that someone who is intelligent and insightful can not only perform intelligent activities, but can explain how and why they think as they do.

Although IML theory represents a prolegomenon toward competition in the Explanation Game, Meta-AQUA can currently play the Explanation Game to only a limited extent. Because the first-order performance tasks (explanation and understanding) is at the periphery of concern, the game is not entertained with any serious intent at this time. Nonetheless, the focus has been on second-order performance (introspection) that constitutes the key component for any system that must explain its own reasoning or must understand itself in any meaningful way. But, to truly qualify for the competition requires more research than can be finished in a reasonable time. The intelligence game is more complex than what we have described so far.

Descartes had a much more skeptical opinion on the possibility of playing these games than either Turing or Schank. Although Descartes was convinced that replicating intelligence was impossible, he was intrigued by the thought of simulating animal and human behavior and thereby increasing the understanding of their mechanics. Indeed, he formulated the first version of the test more than 300 years before Turing reinvented the game.

And this will not seem strange to those, who, knowing how many different automata or moving machines can be made by the industry of man, without employing in doing so more than a very few parts in comparison with the great multitude of bones, muscles, nerves, arteries, veins, or other parts that are found in the body of each animal.... On the other hand, if there were machines which bore a resemblance to our body and imitated our actions as far as it was morally possible to do so, we should always have two very certain tests by which to recognize that, for all that, they were not real men. The first is, that they could never use speech or other signs as we do when placing our thoughts on record for the benefit of others.... And the second difference is, that although machines can perform certain things as well as or perhaps better than any of us can do, they infallibly fall short in others, by the which means we may discover that they did not act from knowledge, but only from the disposition of their organs. For while reason is a universal instrument which can serve for all contingencies, these organs have the need of some special adaptation for every particular action. From this it follows that it is morally impossible that there should be sufficient diversity in any machine to allow it to act in all the events of life in the same way our reason causes us to act.

—Descartes (1637/1955), pp. 115-116.

Descartes' claim is that language, reasoning and knowledge are all integral to intelligence when separating man from machines. The use of knowledge includes all parts of the taxonomy discussed in Section 13.2, including self-knowledge, an especially human feature. Reasoning includes all of the processes discussed in Section 13.1 including introspection. Furthermore, Crockett (1994) claims that learning is critical for any machine's attempt to pass the Turing Test. The program must learn more than rote memorization of the prior pieces of the dialogue during the test. Even given a limited domain, if the judge adds new or hypothetical information relevant to the domain, the program must be able to incorporate this information with what it already knows and make a reasoned comment on the implications of the information (Crockett, 1994). This unfortunately places us in the untenable position of replicating most of human cognition in order to administer a fair test.

Despite the specifics of the test, it appears that a machine must master both learning and introspection in order for it to successfully compete in either Turing’s or Schank’s exams. Knowledge of the world or specific domains is not enough. Knowledge of the self in context with the world is equally important. Neisser (1993; 1995) argues that humans measure knowledge with respect to the self when in the context of problems and understandings of the environment.<sup>150</sup> An understanding of one’s self would include an acquired self-evaluation. Like Wellman’s (1985) person variables, the machine should learn than it is gifted (or deficient) at certain tasks and be able to use this information in its strategies and in its explanations of its own performance.

Both Descartes and Searle were overtly pessimistic about the modeling endeavor; both Turing and Schank are overly optimistic and wish to make operational the concept of intelligence; I tend toward agnosticism. I do not wish to herald the coming of a revolution that is still far-removed, as have others who have studied the parallel between metacognition in humans and machines. For example in the concluding paragraph of a chapter from an expert-systems textbook, Lenat, Davis, Doyle, Genesereth, Goldstein and Schrobe (1983) proclaim the following:

Once self-description is a reality, the next logical step is self-modification. Small, self-modifying, automatic programming systems have existed for a decade; some large programs that modify themselves in very small ways also exist; and the first large fully self-describing and self-modifying programs are being built just now. The capability of machines have finally exceeded human cognitive capabilities in this dimension; it is now worth supplying and using meta-knowledge in large expert systems. (p. 238)

Which human dimension they believe machines have surpassed is uncertain, but it is surely not along the dimension of expert systems knowing themselves, even when given the limitations of human insight. As for their implied prediction concerning the coming impact of self-description and self-modification, no momentous changes have appeared in the last dozen years due to this research. Rather, people such as Buchanan and Smith (1989) are being far more conservative in their statements, claiming that “Expert-systems have little or no self-knowledge” (p. 186). Indeed, many researchers, from Clancey in the AI community to Chi in the psychology community, now tend to avoid altogether the use of words

---

150. In a discouraging note, Clancey (1994) points out that much of what is related to a concept of the self in humans is not very abstract. Rather, the self appears to be closely aligned with perception, the environment, and physiology. Thus, perhaps a model of both the environment and the physiology of the nervous system is more important to a model of introspection than the research here admits. See also Searle (1992) for related arguments on the importance of the neurophysiological level of understanding the mind and arguments against representation.

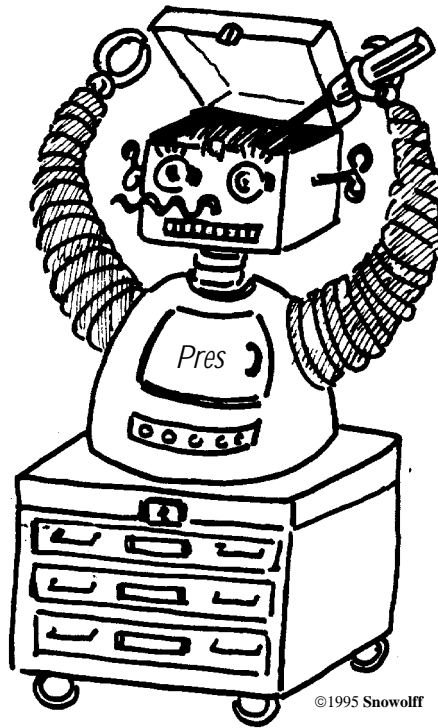
beginning with the prefix “meta” (i.e., the M-words). Rather than run the risk of *the boy who cried wolf too often*, I see room for quiet progress and methodical evaluation before grand predictions can be announced. Knowing fully well the hardships of building even a modest implementation that incorporates some of the levels necessary to model intelligent thought, the goal of concocting a machine of general intelligence is probably beyond our reach, at least in this lifetime.<sup>151</sup>

Regardless of the pragmatics of passing an intelligence test, the business of AI is to develop theories of intelligence, while programs are simply one tool for the research to evaluate these theories (Ram & Jones, 1995). In support of this goal, AI offers the rest of the cognitive science community an ability to think about and to test numerous integrations of processes and knowledge (Schank, 1979, p. 221). Thus, just making progress toward the development of a theory of intelligence and learning and being able to try out different integrations are the important goals; the programs are only the means by which we achieve such an end. In the limit, it does not matter whether Descartes or Turing was correct, nor does it matter whether Schank or Searle had the better position. Research into these issues will benefit an understanding of intelligence and learning either way.

---

151. Of course that is what the pundits said of generally-accessible computing technology just a couple of decades ago. So, if a machine unexpectedly requests that you not shut it off (and you have not explicitly programmed it to say something of that sort), perhaps you should leave it on and run some tests before you conclude that it is just someone’s idea of a joke ;-)







***APPENDICES, REFERENCES AND INDEXES***



## APPENDIX A

### THE DEGREES OF FREEDOM IN LEARNING

IML theory contains a preference for failed experiences, rather than successful ones. The detection of a failure in an input stream is equivalent to self-selection of training examples. Although by filtering examples of successful performance the reasoner may miss some opportunities (and thus bias what can be learned), the input bias trade-off focuses the learner on examples that may require less inference and that guarantee something worth learning exists.<sup>152</sup> That is, failure allows the reasoner to limit search to the space of problems that *do* occur, rather than the much larger space of problems that *may* occur (Hammond, Converse, Marks, & Seifert, 1993). If the reasoner has perfect knowledge, no failure can ever occur (oracles by definition do not fail); thus, failure implies a flaw in knowledge, whereas successful examples may or may not contain any useful lessons.

For failure to occur, the learning system must be associated with some performance task. In the simplest case, the task may be attribute prediction or classification. For example, when a decision tree misclassifies an instance, ID3 (Quinlan, 1986) uses the instance for learning. In general, a failure-driven approach to learning and reasoning concentrates on contradictions, impasses, false expectations, surprises, and unexpected successes during the performance task to indicate when attention is warranted.

Many systems invest too much computational overhead in evaluating examples that, in ordinary performance situations, provide few or no useful opportunities to learn. For example, PRODIGY (Minton, 1990) has no input bias. As a consequence, it learns from every input and then must delete useless knowledge. Alternatively, desJardins' (1992) PAGODA uses a given input's expected utility of predicting features in the environment to filter input examples. Like Meta-AQUA, the system uses a goal-directed learning approach to formulate a set of learning goals that direct and guide the system's learning. However, Meta-AQUA uses an explanation of the system's own failure to generate these goals, while

---

152. If the reasoner mistakenly believes that there was a failure, but actually there was not, then this itself constitutes a failure from which to learn. The challenge in such a case is to detect the actual failure.

PAGODA uses the expected utility of the input. But it is more tractable to let failure feedback from the environment filter the input for useful candidates for goal formulation, rather than calculating the utility of all instances, because fewer instances exist on which to perform computation. Moreover, learning will be simpler in the remaining examples because, as will be shown below, fewer degrees of freedom generally exist for blame assignment when learning from failure than for credit assignment when learning from success.

To illustrate the utility of failure-driven bias, consider the following. During the Persian Gulf oil embargo of Iran, a tragic event occurred that resulted in the death of innocent civilians.<sup>153</sup> The *USS Vincennes* shot down an Iranian commercial airliner after an engagement with Iranian gunboats on July 3, 1988. On the basis of conflicting information, the captain of the *Vincennes* mistook the airliner for an enemy F-14 fighter aircraft and ordered it shot down. Although the incident was controversial, an official investigation concluded that the captain acted in a proper manner given the rules of engagement and the circumstances under which the captain made such a decision.

Instead of this incident being simply a negative example of the category F-14, let us propose three classifiers: one represents the concept “friendly target,” another recognizes “neutral targets,” and a third classifies “enemy targets.” Let us also assume for the sake of simplicity that the friendly-target concept returns negative because of no electronic signature. The remaining two concepts return a value and a confidence level. The captain’s quandary stems from the low confidence returned by the positive identification from the enemy classifier, along with an equally low confidence for the negative classification of neutral aircraft. Given no noise in the data and that an unambiguous result occurs (no possibility of both true or both negative), Table 15 summarizes the possible explanations for answers to the question “Is the reported plane neutral?”

In both failed cases (the shaded cells: false positive and miss), there is only one possibility. If, as in the actual incident, there is a miss (i.e., the actual answer is positive but the expected outcome is negative), then the concept of neutral plane must be overly specialized, since it rejects a positive example; whereas the concept of enemy plane must be overly general since it accepts a negative example. Blame assignment for the converse case, that of a false positive, is equally unambiguous. If the concept of neutral plane mistakenly recognizes an example of an enemy target, then it must be overly general; and at the same time, if the classifier of enemy planes rejects the same example, then the concept must be overly specialized.

---

153. Details of this incident are taken from Thagard (1992).

Table 15: Is the plane neutral? Possible causes<sup>a</sup>

	Neg	Pos
Pos	<b>False Positive</b> $\bar{N} \wedge \underline{E}$ <b>(crew dies)</b>	<b>Hit</b> $\bar{N} \wedge \underline{E}$ $N \wedge E$ $\bar{N} \wedge E$ $N \wedge \underline{E}$
Neg	<b>Correct Rejection</b> $\underline{N} \wedge \bar{E}$ $N \wedge E$ $\underline{N} \wedge E$ $N \wedge \bar{E}$	<b>Miss</b> $\underline{N} \wedge \bar{E}$ <b>(innocents die)</b>

a. N= Neutral target; E = Enemy target; Overscore = overly general; Underscore = overly specialized; Light shading = failed prediction.

In successful examples of performance, many more degrees of freedom exist with which to do credit assignment. In positive identifications of neutral aircraft, in which the neutral classifier returns true and the enemy classifier returns false, both classifiers may still be incorrect in general. That is, although a neutral classifier may be overly general, it can still return true on all positive examples. Likewise, the enemy classifier can be overly specialized and still reject all negative examples. Simply because a particular target is correctly identified, we do not have much information as to the classifier's overall performance. In the case of correct rejection, an overly specialized neutral classifier may still reject a particular enemy aircraft, and even though the enemy classifier may properly accept a particular enemy example, it may still be overly general and succeed. Success gives little information as opposed to failure.

Failure-driven input bias is limited, however. Although failure may constrain learning, some systems may not be able to use this fact because a particular inductive policy (the strategy used to make bias choices based on the underlying assumptions of the domain) may influence a learning system toward certain results (Provost & Buchanan, 1992). Provost and Buchanan show that inductive policies can bias a learner toward speed of acquisition rather than accuracy (when time is a limited resource, for example) or toward accuracy instead of speed (when safety is a high priority). Likewise, in the *Vincennes* scenario, even though failure may facilitate learning, life-critical tasks require that the performance system not choose a course that results in failed examples. The approach of LEX (Mitchell, Utgoff, & Banerji, 1983), which generates learning examples on the basis of their expected utility, irrespective of any inductive policy, is unacceptable. The crew of the *Vincennes* strove for hits and correct rejections despite the fact that much could be learned from examples like the unfortunate incident (miss) that did occur. The consequences of both false positives and misses require an inductive policy that biases the performance system toward accuracy and away from learning optimization.





## APPENDIX B

### META-AQUA OUTPUT IN STORY UNDERSTANDING MODE

Recreate story T6585 by calling (re-run-story 'T6585)

One day Elvis was bored. Elvis pushed cupboard-door away from the cupboard1. The cupboard1 was open. He took the pipel from the cupboard1. He had the pipel. The cupboard1 didn't have the pipel. He pushed cupboard-door to the cupboard1. The cupboard1 wasn't open. He pushed fridge-door away from the fridgel. The fridgel was open. He took the ganjal from the fridgel. He had the ganjal. The fridgel didn't have the ganjal. He pushed fridge-door to the fridgel. The fridgel wasn't open. He poured the ganjal into the pipel. The pipel was filled with the ganjal. He took the lighter1 from the table2. He had the lighter1. The table2 didn't have the lighter1. He pushed the lighter1. The lighter1 was on. He moved the lighter1 to the ganjal. The ganjal was burning. He pushed the lighter1. The lighter1 wasn't on. He smoked the ganjal. The pipel wasn't filled with the ganjal. The pipel was dirty. He exhaled the smokel into the airl. He pushed hot-faucet-handle away from the hot-faucet. The hot-faucet was flowing. He moved the pipel to the hot-faucet. The pipel wasn't dirty. He pushed hot-faucet-handle to the hot-faucet. The hot-faucet wasn't flowing. He smoked the ganjal because he didn't want to be drug-withdrawing.

--- The End ---

Converting Tale-Spin cds into Meta-AQUA frames.

Initialize Memory.

Number concepts to examine for goal generation: 46.

Spawn goal #1 #2 #3 #6 #7 #8 #9 #10 #11 #12 #13 #14 #15 #18 #19 #20 #21  
#26 #27 #28 #29 #30 #35 #36 #37 #38 #39 #44 #45 #46.

Number of sentences to understand: 31.

Begin Meta-AQUA.

Input Structure: PROPEL.6586

"Elvis pushed cupboard-door away from the cupboard1."

Current Sub-Goal:

Identify interesting concepts in PROPEL.6586

PROPEL.6586 is not a very interesting concept.

Skimming . . .

Checking for match...

Input Structure: OPEN.6593

"The cupboard1 was open."

Current Sub-Goal:

Identify interesting concepts in OPEN.6593

OPEN.6593 is not a very interesting concept.

Skimming . . .

Checking for match...

Input Structure: ATRANS.6596

"He took the pipel from the cupboard1."

Current Sub-Goal:

Identify interesting concepts in ATRANS.6596

ATRANS.6596 is not a very interesting concept.

Skimming . . .

Checking for match...

Input Structure: PROPEL.6616

"He pushed cupboard-door to the cupboard1."

Current Sub-Goal:

Identify interesting concepts in PROPEL.6616

PROPEL.6616 is not a very interesting concept.

Skimming . . .

Checking for match...

Instantiating script SMOKING-SCRIPT.2546!!

(PROPEL.2645 ATRANS.2633 PROPEL.2650)

(PROPEL.6586 OPEN.6593 ATRANS.6596 PROPEL.6616)

Unify story concept PROPEL.6586 with scene PROPEL.2645.

(ATrans.2633 PROPEL.2650)

(ATrans.6596 PROPEL.6616)

Unify story concept ATRANS.6596 with scene ATRANS.2633.

(PROPEL.2650)

(PROPEL.6616)

Unify story concept PROPEL.6616 with scene PROPEL.2650.

NIL

Input Structure: OPEN.6623

"The cupboard1 wasn't open."

Current Sub-Goal:

Identify interesting concepts in OPEN.6623

OPEN.6623 is not a very interesting concept.

Skimming . . .

Input Structure: PROPEL.6626

"He pushed fridge-door away from the fridgel."

Current Sub-Goal:

Identify interesting concepts in PROPEL.6626

PROPEL.6626 is not a very interesting concept.

Skimming . . .

Matched story concept PROPEL.6626 with scene PROPEL.9023.

Unify story concept PROPEL.6626 with scene PROPEL.9023.

Lazy unification replacing CUPBOARD.1011 with sibling FRIDGE.1013.

Lazy unification replacing CUPBOARD.1011 with sibling FRIDGE.1013.

Will try to understand script inference OPEN-CONTAINER.9003.

Inferred Structure: OPEN-CONTAINER.9003

"The actor opens the container."

Current Sub-Goal:

Identify interesting concepts in OPEN-CONTAINER.9003

OPEN-CONTAINER.9003 is not a very interesting concept.

Skimming . . .

Lazy unification replacing FRIDGE.1013 with sibling CUPBOARD.1011.

Matched story concept OPEN-CONTAINER.9003 with scene OPEN-CONTAINER.9369.

Unify story concept OPEN-CONTAINER.9003 with scene OPEN-CONTAINER.9369.

Will try to understand script inference GAIN-CONTROL-OF-CONTAINED-OBJECT.9317.

Inferred Structure: GAIN-CONTROL-OF-CONTAINED-OBJECT.9317

NIL

Current Sub-Goal:

Identify interesting concepts in GAIN-CONTROL-OF-CONTAINED-OBJECT.9317

GAIN-CONTROL-OF-CONTAINED-OBJECT.9317 is not a very interesting concept.

Skimming . . .

Matched story concept GAIN-CONTROL-OF-CONTAINED-OBJECT.9317 with scene GAIN-CONTROL-OF-CONTAINED-OBJECT.9855.

Unify story concept GAIN-CONTROL-OF-CONTAINED-OBJECT.9317 with scene GAIN-CONTROL-OF-CONTAINED-OBJECT.9855.

Will try to understand script inference SMOKING-SCRIPT.2546.

Inferred Structure: SMOKING-SCRIPT.2546

NIL

Current Sub-Goal:

Identify interesting concepts in SMOKING-SCRIPT.2546

SMOKING-SCRIPT.2546 is not a very interesting concept.

Skimming . . .

Matched story concept SMOKING-SCRIPT.2546 with scene SMOKING-SCRIPT.2546.

Unify story concept SMOKING-SCRIPT.2546 with scene SMOKING-SCRIPT.2546.

Input Structure: OPEN.6633

"The fridgel was open."

Current Sub-Goal:

Identify interesting concepts in OPEN.6633

OPEN.6633 is not a very interesting concept.

Skimming . . .

Input Structure: ATRANS.6636

"He took the ganjal from the fridgel."

Current Sub-Goal:

Identify interesting concepts in ATRANS.6636

ATrans.6636 is not a very interesting concept.

Skimming . . .

Input Structure: PROPEL.6656

"He pushed fridge-door to the fridgel."

Current Sub-Goal:

Identify interesting concepts in PROPEL.6656

PROPEL.6656 is not a very interesting concept.

Skimming . . .

Matched story concept PROPEL.6656 with scene PROPEL.11624.

Unify story concept PROPEL.6656 with scene PROPEL.11624.

Will try to understand script inference OPEN-CONTAINER.11604.

Inferred Structure: OPEN-CONTAINER.11604

"The actor opens the container."

Current Sub-Goal:

Identify interesting concepts in OPEN-CONTAINER.11604

OPEN-CONTAINER.11604 is not a very interesting concept.

Skimming . . .

Matched story concept OPEN-CONTAINER.11604 with scene OPEN-CONTAINER.11985.

Unify story concept OPEN-CONTAINER.11604 with scene OPEN-CONTAINER.11985.

Will try to understand script inference GAIN-CONTROL-OF-CONTAINED-OBJECT.11933.

Inferred Structure: GAIN-CONTROL-OF-CONTAINED-OBJECT.11933

NIL

Current Sub-Goal:

Identify interesting concepts in GAIN-CONTROL-OF-CONTAINED-OBJECT.11933

GAIN-CONTROL-OF-CONTAINED-OBJECT.11933 is not a very interesting concept.

Skimming . . .

Matched story concept GAIN-CONTROL-OF-CONTAINED-OBJECT.11933 with scene GAIN-CONTROL-OF-CONTAINED-OBJECT.12482.

Unify story concept GAIN-CONTROL-OF-CONTAINED-OBJECT.11933 with scene GAIN-CONTROL-OF-CONTAINED-OBJECT.12482.

Will try to understand script inference SMOKING-SCRIPT.2546.

Inferred Structure: SMOKING-SCRIPT.2546

NIL

Current Sub-Goal:

Identify interesting concepts in SMOKING-SCRIPT.2546

SMOKING-SCRIPT.2546 is not a very interesting concept.

Skimming . . .

Matched story concept SMOKING-SCRIPT.2546 with scene SMOKING-SCRIPT.2546.

Unify story concept SMOKING-SCRIPT.2546 with scene SMOKING-SCRIPT.2546.

Input Structure: OPEN.6663

"The fridgel wasn't open."

Current Sub-Goal:

Identify interesting concepts in OPEN.6663

OPEN.6663 is not a very interesting concept.

Skimming . . .

Input Structure: TILT.6666

"He poured the ganjal into the pipel."

Current Sub-Goal:

Identify interesting concepts in TILT.6666

TILT.6666 is not a very interesting concept.

Skimming . . .

Matched story concept TILT.6666 with scene TILT.13992.

Unify story concept TILT.6666 with scene TILT.13992.

Will try to understand script inference FILL-PIPE.13917.

Inferred Structure: FILL-PIPE.13917

"Actor fills the pipe."

Current Sub-Goal:

Identify interesting concepts in FILL-PIPE.13917



FILL-PIPE.13917 is not a very interesting concept.

Skimming . . .

Matched story concept FILL-PIPE.13917 with scene FILL-PIPE.14279.

Unify story concept FILL-PIPE.13917 with scene FILL-PIPE.14279.

Will try to understand script inference SMOKING-SCRIPT.2546.

Inferred Structure: SMOKING-SCRIPT.2546

NIL

Current Sub-Goal:

Identify interesting concepts in SMOKING-SCRIPT.2546

SMOKING-SCRIPT.2546 is not a very interesting concept.

Skimming . . .

Matched story concept SMOKING-SCRIPT.2546 with scene SMOKING-SCRIPT.2546.

Unify story concept SMOKING-SCRIPT.2546 with scene SMOKING-SCRIPT.2546.

Input Structure: FILLED.6674

"The pipel was filled with the ganjal."

Current Sub-Goal:

Identify interesting concepts in FILLED.6674

FILLED.6674 is not a very interesting concept.

Skimming . . .

Input Structure: ATRANS.6677

"He took the lighter1 from the table2."

Current Sub-Goal:

Identify interesting concepts in ATRANS.6677

ATRANS.6677 is not a very interesting concept.

Skimming . . .

Input Structure: PROPEL.6697

"He pushed the lighter1."

Current Sub-Goal:

Identify interesting concepts in PROPEL.6697

PROPEL.6697 is not a very interesting concept.

Skimming . . .

Matched story concept PROPEL.6697 with scene PROPEL.16098.

Unify story concept PROPEL.6697 with scene PROPEL.16098.

Will try to understand script inference TURN-ON.16085.

Inferred Structure: TURN-ON.16085

NIL

Current Sub-Goal:

Identify interesting concepts in TURN-ON.16085

TURN-ON.16085 is not a very interesting concept.

Skimming . . .

Matched story concept TURN-ON.16085 with scene TURN-ON.16478.

Unify story concept TURN-ON.16085 with scene TURN-ON.16478.

Will try to understand script inference LIGHT-OBJECT.16457.

Inferred Structure: LIGHT-OBJECT.16457

NIL

Current Sub-Goal:

Identify interesting concepts in LIGHT-OBJECT.16457

LIGHT-OBJECT.16457 is not a very interesting concept.

Skimming . . .

Matched story concept LIGHT-OBJECT.16457 with scene LIGHT-OBJECT.16929.

Unify story concept LIGHT-OBJECT.16457 with scene LIGHT-OBJECT.16929.

Will try to understand script inference SMOKE-PIPE.16857.

Inferred Structure: SMOKE-PIPE.16857

"Actor smokes a pipe."

Current Sub-Goal:

Identify interesting concepts in SMOKE-PIPE.16857

SMOKE-PIPE.16857 is not a very interesting concept.

Skimming . . .

Matched story concept SMOKE-PIPE.16857 with scene SMOKE-PIPE.17424.

Unify story concept SMOKE-PIPE.16857 with scene SMOKE-PIPE.17424.

Will try to understand script inference SMOKING-SCRIPT.2546.

Inferred Structure: SMOKING-SCRIPT.2546

NIL

Current Sub-Goal:

Identify interesting concepts in SMOKING-SCRIPT.2546

SMOKING-SCRIPT.2546 is not a very interesting concept.

Skimming . . .

Matched story concept SMOKING-SCRIPT.2546 with scene SMOKING-SCRIPT.2546.

Unify story concept SMOKING-SCRIPT.2546 with scene SMOKING-SCRIPT.2546.

Input Structure: TURNED-ON.6702

"The lighter1 was on."

Current Sub-Goal:

Identify interesting concepts in TURNED-ON.6702

TURNED-ON.6702 is not a very interesting concept.

Skimming . . .

Input Structure: PTRANS.6705

"He moved the lighter1 to the ganjal. "

Current Sub-Goal:

Identify interesting concepts in PTRANS.6705

PTRANS.6705 is not a very interesting concept.

Skimming . . .

Matched story concept PTRANS.6705 with scene PTRANS.19013.

Unify story concept PTRANS.6705 with scene PTRANS.19013.

Lazy unification replacing TOBACCO.18918 with sibling MARIJUANA.966.

Will try to understand script inference LIGHT-OBJECT.18988.

Inferred Structure: LIGHT-OBJECT.18988

NIL

Current Sub-Goal:

Identify interesting concepts in LIGHT-OBJECT.18988

LIGHT-OBJECT.18988 is not a very interesting concept.

Skimming . . .

Lazy unification replacing TOBACCO.19401 with sibling MARIJUANA.966.

Matched story concept LIGHT-OBJECT.18988 with scene LIGHT-OBJECT.19471.

Lazy unification replacing TOBACCO.19401 with sibling MARIJUANA.966.

Unify story concept LIGHT-OBJECT.18988 with scene LIGHT-OBJECT.19471.

Will try to understand script inference SMOKE-PIPE.19399.

Inferred Structure: SMOKE-PIPE.19399

"Actor smokes a pipe."

Current Sub-Goal:

Identify interesting concepts in SMOKE-PIPE.19399

SMOKE-PIPE.19399 is not a very interesting concept.

Skimming . . .

Matched story concept SMOKE-PIPE.19399 with scene SMOKE-PIPE.19964.

Unify story concept SMOKE-PIPE.19399 with scene SMOKE-PIPE.19964.

Will try to understand script inference SMOKING-SCRIPT.2546.

Inferred Structure: SMOKING-SCRIPT.2546

NIL

Current Sub-Goal:

Identify interesting concepts in SMOKING-SCRIPT.2546

SMOKING-SCRIPT.2546 is not a very interesting concept.

Skimming . . .

Matched story concept SMOKING-SCRIPT.2546 with scene SMOKING-SCRIPT.2546.

Unify story concept SMOKING-SCRIPT.2546 with scene SMOKING-SCRIPT.2546.

Input Structure: BURNING.6712

"The ganjal was burning. "

Current Sub-Goal:

Identify interesting concepts in BURNING.6712

BURNING.6712 is not a very interesting concept.

Skimming . . .

Lazy unification replacing TOBACCO.21193 with sibling MARIJUANA.966.  
Input Structure: PROPEL.6723

"He pushed the lighter1."

Current Sub-Goal:

Identify interesting concepts in PROPEL.6723

PROPEL.6723 is not a very interesting concept.

Skimming . . .

Lazy unification replacing TOBACCO.21457 with sibling MARIJUANA.966.  
Matched story concept PROPEL.6723 with scene PROPEL.21561.

Lazy unification replacing TOBACCO.21457 with sibling MARIJUANA.966.  
Unify story concept PROPEL.6723 with scene PROPEL.21561.

Will try to understand script inference TURN-ON.21548.

Inferred Structure: TURN-ON.21548

NIL

Current Sub-Goal:

Identify interesting concepts in TURN-ON.21548

TURN-ON.21548 is not a very interesting concept.

Skimming . . .

Lazy unification replacing TOBACCO.21847 with sibling MARIJUANA.966.  
Matched story concept TURN-ON.21548 with scene TURN-ON.21938.

Lazy unification replacing TOBACCO.21847 with sibling MARIJUANA.966.  
Unify story concept TURN-ON.21548 with scene TURN-ON.21938.

Will try to understand script inference LIGHT-OBJECT.21917.

Inferred Structure: LIGHT-OBJECT.21917

NIL

Current Sub-Goal:

Identify interesting concepts in LIGHT-OBJECT.21917

LIGHT-OBJECT.21917 is not a very interesting concept.

Skimming . . .

Lazy unification replacing TOBACCO.22315 with sibling MARIJUANA.966.  
Matched story concept LIGHT-OBJECT.21917 with scene LIGHT-OBJECT.22385.

Lazy unification replacing TOBACCO.22315 with sibling MARIJUANA.966.  
 Unify story concept LIGHT-OBJECT.21917 with scene LIGHT-OBJECT.22385.

Will try to understand script inference SMOKE-PIPE.22313.

Inferred Structure: SMOKE-PIPE.22313

"Actor smokes a pipe."

Current Sub-Goal:

Identify interesting concepts in SMOKE-PIPE.22313

SMOKE-PIPE.22313 is not a very interesting concept.

Skimming . . .

Matched story concept SMOKE-PIPE.22313 with scene SMOKE-PIPE.22878.

Unify story concept SMOKE-PIPE.22313 with scene SMOKE-PIPE.22878.

Will try to understand script inference SMOKING-SCRIPT.2546.

Inferred Structure: SMOKING-SCRIPT.2546

NIL

Current Sub-Goal:

Identify interesting concepts in SMOKING-SCRIPT.2546

SMOKING-SCRIPT.2546 is not a very interesting concept.

Skimming . . .

Matched story concept SMOKING-SCRIPT.2546 with scene SMOKING-SCRIPT.2546.

Unify story concept SMOKING-SCRIPT.2546 with scene SMOKING-SCRIPT.2546.

Input Structure: TURNED-ON.6728

"The lighter1 wasn't on."



Current Sub-Goal:

Identify interesting concepts in TURNED-ON.6728

TURNED-ON.6728 is not a very interesting concept.

Skimming . . .

Lazy unification replacing TOBACCO.24107 with sibling MARIJUANA.966.  
Input Structure: INGEST.6731

"He smoked the ganjal."

Current Sub-Goal:

Identify interesting concepts in INGEST.6731

INGEST.6731 is not a very interesting concept.

Skimming . . .

Lazy unification replacing TOBACCO.24376 with sibling MARIJUANA.966.  
Matched story concept INGEST.6731 with scene INGEST.24456.

Lazy unification replacing TOBACCO.24376 with sibling MARIJUANA.966.  
Unify story concept INGEST.6731 with scene INGEST.24456.

Lazy unification replacing TOBACCO.24376 with sibling MARIJUANA.966.  
Lazy unification replacing TOBACCO.24376 with sibling MARIJUANA.966.  
Lazy unification replacing TOBACCO.24376 with sibling MARIJUANA.966.  
Lazy unification replacing TOBACCO.24376 with sibling MARIJUANA.966.

Will try to understand script inference SMOKE-PIPE.24374.

Inferred Structure: SMOKE-PIPE.24374

"Actor smokes a pipe."

Current Sub-Goal:

Identify interesting concepts in SMOKE-PIPE.24374

Anomaly detected: Odd for a MARIJUANA

to be in path (GOAL-SCENE OBJECT) of a SMOKE-PIPE.

Anomaly detected: Odd for a MARIJUANA

to be in path (GOAL-SCENE FROM DOMAIN) of a SMOKE-PIPE.

Anomaly detected: Odd for a MARIJUANA

to be in path (GOAL-SCENE FROM CO-DOMAIN DOMAIN) of a SMOKE-PIPE.

Anomaly detected: Odd for a MARIJUANA

to be in path (GOAL-SCENE TO DOMAIN) of a SMOKE-PIPE.

Posing Question: ACTOR.9965

Why did the ADULT ADULT.24375 perform the SMOKE-PIPE?

Current Sub-Goal:

Generate explanation for why ADULT.24375 decides to perform SMOKE-PIPE.24374

Found explanation(s)

(XP-GOAL-OF-OUTCOME->ACTOR.100).

Explaining concept ACTOR.9965.

Trying explanation XP-GOAL-OF-OUTCOME->ACTOR.25104

on instance role ACTOR.9965.

Explained node ACTOR.9965 successfully

unified with instance.

Asserted node STATE.25108 already known

Explanation is "Actor does action because it achieves a goal the agent desires.".

XP: XP-GOAL-OF-OUTCOME->ACTOR.25104.

Current Sub-Goal:

Verify hypothesis XP-GOAL-OF-OUTCOME->ACTOR.25104

Cannot achieve GOAL.25651 at this time.

Suspend TEST task . . .

Input Structure: FILLED.6737

"The pipel wasn't filled with the ganjal."

Current Sub-Goal:

Identify interesting concepts in FILLED.6737

FILLED.6737 is not a very interesting concept.

Skimming . . .

Lazy unification replacing MARIJUANA.966 with sibling TOBACCO.25838.  
 Lazy unification replacing MARIJUANA.966 with sibling TOBACCO.25838.  
 Lazy unification replacing TOBACCO.25838 with sibling MARIJUANA.966.  
 Lazy unification replacing MARIJUANA.966 with sibling TOBACCO.25838.  
 Lazy unification replacing TOBACCO.25838 with sibling MARIJUANA.966.  
 Input Structure: DIRTY.6740

"The pipel was dirty."

Current Sub-Goal:

Identify interesting concepts in DIRTY.6740

DIRTY.6740 is not a very interesting concept.

Skimming . . .

Lazy unification replacing TOBACCO.26091 with sibling MARIJUANA.966.  
 Lazy unification replacing TOBACCO.26091 with sibling MARIJUANA.966.  
 Input Structure: EXPEL.6743

"He exhaled the smokel into the airl."

Current Sub-Goal:

Identify interesting concepts in EXPEL.6743

EXPEL.6743 is not a very interesting concept.

Skimming . . .

Lazy unification replacing TOBACCO.26356 with sibling MARIJUANA.966.  
 Lazy unification replacing TOBACCO.26356 with sibling MARIJUANA.966.  
 Matched story concept EXPEL.6743 with scene EXPEL.26440.

Lazy unification replacing TOBACCO.26356 with sibling MARIJUANA.966.  
 Lazy unification replacing TOBACCO.26356 with sibling MARIJUANA.966.  
 Unify story concept EXPEL.6743 with scene EXPEL.26440.

Will try to understand script inference SMOKE-PIPE.26354.

Inferred Structure: SMOKE-PIPE.26354

"Actor smokes a pipe."

Current Sub-Goal:

Identify interesting concepts in SMOKE-PIPE.26354

Anomaly detected: Odd for a MARIJUANA

to be in path (GOAL-SCENE FROM CO-DOMAIN DOMAIN) of a SMOKE-PIPE.

Repeating Old Question: ACTOR.9965

Why did the ADULT ADULT.26355 perform the SMOKE-PIPE?

Current Sub-Goal:

Generate explanation for why ADULT.26355 decides to perform SMOKE-PIPE.26354

Found explanation(s)

(XP-GOAL-OF-OUTCOME->ACTOR.100).

Explaining concept ACTOR.9965.

Trying explanation XP-GOAL-OF-OUTCOME->ACTOR.27061

on instance role ACTOR.9965.

Unification of truth values IN.0 and HYPOTHESIZED.0

Explained node ACTOR.9965 successfully

unified with instance.

Explanation is "Actor does action because it achieves a goal the agent desires.".

XP: XP-GOAL-OF-OUTCOME->ACTOR.27061.

Current Sub-Goal:

Verify hypothesis XP-GOAL-OF-OUTCOME->ACTOR.27061

Cannot achieve GOAL.27849 at this time.

Suspend TEST task . . .

Input Structure: PROPEL.6749

"He pushed hot-faucet-handle away from the hot-faucet."

Current Sub-Goal:

Identify interesting concepts in PROPEL.6749

PROPEL.6749 is not a very interesting concept.

Skimming . . .

Lazy unification replacing TOBACCO.28068 with sibling MARIJUANA.966.  
 Lazy unification replacing TOBACCO.28068 with sibling MARIJUANA.966.  
 Matched story concept PROPEL.6749 with scene PROPEL.28182.

Lazy unification replacing TOBACCO.28068 with sibling MARIJUANA.966.  
 Lazy unification replacing TOBACCO.28068 with sibling MARIJUANA.966.  
 Unify story concept PROPEL.6749 with scene PROPEL.28182.

Will try to understand script inference WASH-ITEM.28086.

Inferred Structure: WASH-ITEM.28086

NIL

Current Sub-Goal:

Identify interesting concepts in WASH-ITEM.28086

WASH-ITEM.28086 is not a very interesting concept.

Skimming . . .

Lazy unification replacing TOBACCO.28710 with sibling MARIJUANA.966.  
 Lazy unification replacing TOBACCO.28710 with sibling MARIJUANA.966.  
 Matched story concept WASH-ITEM.28086 with scene WASH-ITEM.28728.

Lazy unification replacing TOBACCO.28710 with sibling MARIJUANA.966.  
 Lazy unification replacing TOBACCO.28710 with sibling MARIJUANA.966.  
 Unify story concept WASH-ITEM.28086 with scene WASH-ITEM.28728.

Will try to understand script inference SMOKING-SCRIPT.2546.

Inferred Structure: SMOKING-SCRIPT.2546

NIL

Input triggers reminding of old question:

(ACTOR.27070)

New input does not help answer old question.

Current Sub-Goal:

Identify interesting concepts in SMOKING-SCRIPT.2546

Anomaly detected: Odd for a MARIJUANA

to be in path (GOAL-SCENE FROM CO-DOMAIN DOMAIN) of a SMOKE-PIPE.

Posing Question: ACTOR.9932

Why did the ADULT ADULT.28709 perform the SMOKING-SCRIPT?

Current Sub-Goal:

Generate explanation for why ADULT.28709 decides to perform SMOKING-SCRIPT.2546

No explanation found.

No case found.

Cannot achieve GOAL.30021 at this time.

Suspend GENERATE task . . .

Input Structure: FLOWING.6756

"The hot-faucet was flowing."

Current Sub-Goal:

Identify interesting concepts in FLOWING.6756

FLOWING.6756 is not a very interesting concept.

Skimming . . .

Lazy unification replacing TOBACCO.30212 with sibling MARIJUANA.966.  
 Lazy unification replacing TOBACCO.30212 with sibling MARIJUANA.966.  
 Input Structure: PTRANS.6759

"He moved the pipel to the hot-faucet."

Current Sub-Goal:

Identify interesting concepts in PTRANS.6759

PTRANS.6759 is not a very interesting concept.

Skimming . . .

Lazy unification replacing TOBACCO.30488 with sibling MARIJUANA.966.  
 Lazy unification replacing TOBACCO.30488 with sibling MARIJUANA.966.  
 Matched story concept PTRANS.6759 with scene PTRANS.30607.

Lazy unification replacing TOBACCO.30488 with sibling MARIJUANA.966.  
 Lazy unification replacing TOBACCO.30488 with sibling MARIJUANA.966.  
 Unify story concept PTRANS.6759 with scene PTRANS.30607.

Will try to understand script inference WASH-ITEM.30506.

Inferred Structure: WASH-ITEM.30506

NIL

Current Sub-Goal:

Identify interesting concepts in WASH-ITEM.30506

WASH-ITEM.30506 is not a very interesting concept.

Skimming . . .

Lazy unification replacing TOBACCO.31129 with sibling MARIJUANA.966.  
 Lazy unification replacing TOBACCO.31129 with sibling MARIJUANA.966.  
 Matched story concept WASH-ITEM.30506 with scene WASH-ITEM.31147.



Lazy unification replacing TOBACCO.31129 with sibling MARIJUANA.966.  
 Lazy unification replacing TOBACCO.31129 with sibling MARIJUANA.966.  
 Unify story concept WASH-ITEM.30506 with scene WASH-ITEM.31147.

Will try to understand script inference SMOKING-SCRIPT.2546.

Inferred Structure: SMOKING-SCRIPT.2546

NIL

Current Sub-Goal:

Identify interesting concepts in SMOKING-SCRIPT.2546

Anomaly detected: Odd for a MARIJUANA

to be in path (GOAL-SCENE FROM CO-DOMAIN DOMAIN) of a SMOKE-PIPE.

Repeating Old Question: ACTOR.9932

Why did the ADULT ADULT.31128 perform the SMOKING-SCRIPT?

Current Sub-Goal:

Generate explanation for why ADULT.31128 decides to perform SMOKING-SCRIPT.2546

No explanation found.

No case found.

Cannot achieve GOAL.32426 at this time.

Suspend GENERATE task . . .

Input Structure: DIRTY.6766

"The pipe1 wasn't dirty."

Current Sub-Goal:

Identify interesting concepts in DIRTY.6766

DIRTY.6766 is not a very interesting concept.

Skimming . . .

Lazy unification replacing TOBACCO.32612 with sibling MARIJUANA.966.  
 Lazy unification replacing TOBACCO.32612 with sibling MARIJUANA.966.  
 Input Structure: PROPEL.6777

"He pushed hot-faucet-handle to the hot-faucet."

Current Sub-Goal:

Identify interesting concepts in PROPEL.6777

PROPEL.6777 is not a very interesting concept.

Skimming . . .

Lazy unification replacing TOBACCO.32902 with sibling MARIJUANA.966.  
 Lazy unification replacing TOBACCO.32902 with sibling MARIJUANA.966.  
 Matched story concept PROPEL.6777 with scene PROPEL.33016.

Lazy unification replacing TOBACCO.32902 with sibling MARIJUANA.966.  
 Lazy unification replacing TOBACCO.32902 with sibling MARIJUANA.966.  
 Unify story concept PROPEL.6777 with scene PROPEL.33016.

Will try to understand script inference WASH-ITEM.32920.

Inferred Structure: WASH-ITEM.32920

NIL

Current Sub-Goal:

Identify interesting concepts in WASH-ITEM.32920

WASH-ITEM.32920 is not a very interesting concept.

Skimming . . .

Lazy unification replacing TOBACCO.33555 with sibling MARIJUANA.966.  
 Lazy unification replacing TOBACCO.33555 with sibling MARIJUANA.966.  
 Matched story concept WASH-ITEM.32920 with scene WASH-ITEM.33573.

Lazy unification replacing TOBACCO.33555 with sibling MARIJUANA.966.  
 Lazy unification replacing TOBACCO.33555 with sibling MARIJUANA.966.  
 Unify story concept WASH-ITEM.32920 with scene WASH-ITEM.33573.

Will try to understand script inference SMOKING-SCRIPT.2546.

Inferred Structure: SMOKING-SCRIPT.2546

NIL

Current Sub-Goal:

Identify interesting concepts in SMOKING-SCRIPT.2546

Anomaly detected: Odd for a MARIJUANA

to be in path (GOAL-SCENE FROM CO-DOMAIN DOMAIN) of a SMOKE-PIPE.

Repeating Old Question: ACTOR.9932

Why did the ADULT ADULT.33554 perform the SMOKING-SCRIPT?

Current Sub-Goal:

Generate explanation for why ADULT.33554 decides to perform SMOKING-SCRIPT.2546

No explanation found.

No case found.

Cannot achieve GOAL.34869 at this time.

Suspend GENERATE task . . .

Input Structure: FLOWING.6784

"The hot-faucet wasn't flowing."

Current Sub-Goal:

Identify interesting concepts in FLOWING.6784

FLOWING.6784 is not a very interesting concept.

Skimming . . .

Lazy unification replacing TOBACCO.35060 with sibling MARIJUANA.966.

Lazy unification replacing TOBACCO.35060 with sibling MARIJUANA.966.

Input Structure: XP-GOAL-OF-OUTCOME->ACTOR.6787

"He smoked the ganjal because he didn't want to be withdrawing."

Input triggers reminding of old question:

(ACTOR.27070 ACTOR.27070 ACTOR.27070)

Lazy unification replacing SMOKE-PIPE.35058 with sibling INGEST.6802.

Lazy unification replacing SMOKE-PIPE.35058 with sibling INGEST.6802.

Lazy unification replacing SMOKE-PIPE.35058 with sibling INGEST.6802.

Lazy unification replacing SMOKE-PIPE.35058 with sibling INGEST.6802.

Lazy unification replacing SMOKE-PIPE.35058 with sibling INGEST.6802.

Lazy unification replacing SMOKE-PIPE.35058 with sibling INGEST.6802.

Lazy unification replacing SMOKE-PIPE.35058 with sibling INGEST.6802.

Lazy unification replacing SMOKE-PIPE.35058 with sibling INGEST.6802.

Question ACTOR.27070 successfully answered

with matching input.

Found relevant input for verifying hypothesis: XP-GOAL-OF-OUTCOME->ACTOR.6787

Comparison strategy selected

Compare strategy applied to evidence XP-GOAL-OF-OUTCOME->ACTOR.6787 for hypothesis XP-GOAL-OF-OUTCOME->ACTOR.6787

Current Sub-Goal:

Review reasoning trace TRACE-META-XP.35229

LEARNING PHASE . . .

Found explanation(s)

(IMXP-ANOMALY-EXPLAINED.351).

Explaining concept SUCCESSFUL-PREDICTION.35207.

Trying explanation IMXP-ANOMALY-EXPLAINED.35281

on instance role SUCCESSFUL-PREDICTION.35207.

XP IMXP-ANOMALY-EXPLAINED.35281 does not apply to

instance SUCCESSFUL-PREDICTION.35467 because missing knowledge

(INCORPORATION-FAILURE.35424 ANOMALY.35303)

Explanation IMXP-ANOMALY-EXPLAINED.35281 does not apply to SUCCESSFUL-PREDICTION.35207.

"No name"

Explanation is "No name".

XP: IMXP-ANOMALY-EXPLAINED.35281.

(MODEL (FRAME-LIST (VALUE (GOAL.35214))))

(MODEL (FRAME-LIST (VALUE (GOAL.35894))))

Blame assignment has produced explanation of reasoning failure:

IMXP-ANOMALY-EXPLAINED.35281

Deciding what to learn.

Posting the following learning goals:

(KNOWLEDGE-RECONCILIATION-GOAL.36472)

with priorities (EIGHT.0)

Selecting learning strategies.

The following algorithms are selected:

(LEARNING-PROCESS.36467)

Executing strategies.

Perform abstraction to (TOBACCO)

on conceptual definition of SMOKE-PIPE.

Done.

Time to completion: 25.683332 minutes.

## APPENDIX C

### STORY LISTINGS FOR RUN NUMBER FOUR

#### Story number 1 . . .

One day ...MOM was BORED. Mom asked Karen, "Would you push the balloon3 to me away from you?" She asked her, "Would I take the balloon3 from me?" She pushed cupboard-door away from the cupboard1. The cupboard1 was open. She took the balloon3 from the cupboard1. She had the balloon3. The cupboard1 didn't have the balloon3. She pushed cupboard-door to the cupboard1. The cupboard1 wasn't open. She pushed light-switch1. The light1 was on. She exhaled the air2 into the balloon3. The balloon3 was inflated. She tied the balloon3. The balloon3 was sealed. The phonel was ringing. Dad picked up phone-receiver1. The phonel wasn't ringing. He had phone-receiver1. He let go of phone-receiver1. He didn't have phone-receiver1. She took the balloon3 from her. She had the balloon3. She didn't have the balloon3. She went to outside. Karen went to outside. Mom played with the balloon3. Karen pushed the balloon3 to Mom away from her. Mom had the balloon3. Karen didn't have the balloon3. Mom pushed the balloon3 to the window1. Karen picked up the balloon3. She had the balloon3. She pushed the balloon3 to Mom away from her. Mom had the balloon3. Karen didn't have the balloon3. Mom played with the balloon3 because she didn't want to be bored.

--- The End ---

#### Story number 2 . . .

One day ...MOM was BORED. Mom asked Karen, "Would you push the ball1 to me away from you?" Karen went to the garage. She picked up the ball1. She had the ball1. She went to outside. Mom went to outside. She played with the ball1. Karen pushed the ball1 to the rose-bush2. Mom picked up the ball1. She had the ball1. She hit the ball1. She hit the ball1 because she wanted to move the ball1 to Karen. Karen hit the ball1. She hit the ball1 because she wanted to move the ball1 to Mom. Mom pushed the ball1 to the grass away from her. She didn't have the ball1. She picked up the ball1. She had the ball1. She pushed the ball1 to the rose-bush1. Karen picked up the ball1. She had the ball1. She pushed the ball1 to the calla-lilly1. Mom picked up the ball1. She had the ball1.

She played with the ball1 because she didn't want to be bored.

--- The End ---

### Story number 3 . . .

One day ...ELVIS was JONESING. Elvis pushed cupboard-door away from the cupboard1. The cupboard1 was open. He took the pipe2 from the cupboard1. He had the pipe2. The cupboard1 didn't have the pipe2. He pushed cupboard-door to the cupboard1. The cupboard1 wasn't open. He took the ganjal from the rug1. He had the ganjal. The rug1 didn't have the ganjal. The phonel was ringing. Dad picked up phone-receiver1. The phonel wasn't ringing. He had phone-receiver1. He let go of phone-receiver1. He didn't have phone-receiver1. Elvis poured the ganjal into the pipe2. The pipe2 was filled with the ganjal. He took the lighter1 from the table2. He had the lighter1. The table2 didn't have the lighter1. He pushed the lighter1. The lighter1 was on. Police-and-dogs arrived. Officer1 went to outside. The police-dog1 went to outside. He pushed door-bell-switch1. The door-bell1 was ringing. He didn't push door-bell-switch1. The door-bell1 wasn't ringing. He went to the kitchen. The police-dog1 went to the kitchen. The police-dog1 went to Elvis. The police-dog1 sniffed Elvis. The police-dog1 barked at Elvis. The police-dog1 was barking. He went to Elvis. He took the ganjal from Elvis. He had the ganjal. Elvis didn't have the ganjal. Officer1 arrested Elvis. He controlled Elvis. He arrested Elvis because he wanted to control Elvis. He went to outside. Elvis went to outside. The police-dog1 went to outside. The police-dog1 barked at him because the police-dog1 detected the ganjal. He had the lighter1. The ganjal couldn't be burning. Officer1 controlled Elvis. Elvis couldn't get near the pipel. He couldn't get the pipel. The ganjal couldn't fill the pipel. He was still jonesing.

--- The End ---

### Story number 4 . . .

One day ...DAD was JONESING. Dad pushed cupboard-door away from the cupboard1. The cupboard1 was open. He took the pipe2 from the cupboard1. He had the pipe2. The cupboard1 didn't have the pipe2. He pushed cupboard-door to the cupboard1. The cupboard1 wasn't open. He took the pipe-tobacco1 from the table1. He had the pipe-tobacco1. The table1 didn't have the pipe-tobacco1. The cat1 pushed the vase2 to the floor1. The vase2 was broken. He poured the pipe-tobacco1 into the pipe2. The pipe2 was filled with the pipe-tobacco1. He took the lighter1 from the table2. He had the lighter1. The table2 didn't have the lighter1. He pushed the lighter1. The lighter1 was on. He moved the lighter1 to the



pipe-tobacco1. The pipe-tobacco1 was burning. The police arrived. Officer1 went to outside1. The dog1 barked. The dog1 was barking. He pushed door-bell-switch1. The door-bell1 was ringing. He didn't push door-bell-switch1. The door-bell1 wasn't ringing. Mom pushed light-switch1. The light1 was on. He went to the kitchen. He asked Elvis whether Elvis would give him the ganjal. Elvis took the ganjal from the rug1. He had the ganjal. The rug1 didn't have the ganjal. He gave officer1 the ganjal. Officer1 had the ganjal. Elvis didn't have the ganjal. Officer1 arrested Elvis. He controlled Elvis. He went to outside. Elvis went to outside. Officer1 arrested Elvis because he wanted to control Elvis. Dad pushed the lighter1. The lighter1 wasn't on. The phone1 was ringing. She picked up phone-receiver1. The phone1 wasn't ringing. She had phone-receiver1. She let go of phone-receiver1. She didn't have phone-receiver1. He smoked the pipe-tobacco1. The pipe2 wasn't filled with the pipe-tobacco1. The pipe2 was dirty. He exhaled the smoke1 into the air1. He pushed hot-faucet-handle away from the hot-faucet. The hot-faucet was flowing. He moved the pipe2 to the hot-faucet. The pipe2 wasn't dirty. He pushed hot-faucet-handle to the hot-faucet. The hot-faucet wasn't flowing. He smoked the pipe-tobacco1 because he didn't want to be withdrawing.

--- The End ---

#### Story number 5 . . .

One day ...MOM was BORED. Mom asked Karen, "Would you push the ball2 to me away from you?" She pushed light-switch1. The light1 was on. Karen went to the garage. The phone1 was ringing. Dad picked up phone-receiver1. The phone1 wasn't ringing. He had phone-receiver1. He let go of phone-receiver1. He didn't have phone-receiver1. She picked up the ball2. She had the ball2. She went to outside. Mom went to outside. She played with the ball2. Karen pushed the ball2 to the window1. The window1 was sharp. The window1 was shattered. The cat1 pushed the vase2 to the floor1. The vase2 was broken. Mom picked up the ball2. She had the ball2. She pushed the ball2 to the calla-lilly2. Karen picked up the ball2. She had the ball2. Mom played with the ball2 because she didn't want to be bored.

--- The End ---

#### Story number 6 . . .

One day ...OFFICER1 was CONCERNED. Officer1 went to the kitchen. He asked Elvis whether Elvis would give him the ganjal. Elvis took the ganjal from the vase3. He had the ganjal. The vase3 didn't have the ganjal. He gave officer1 the ganjal. Officer1 had the ganjal. Elvis didn't have the ganjal. Officer1 arrested Elvis. He controlled Elvis. He went to outside. The cat1 pushed the vase2 to the floor1. The vase2 was broken. Mom pushed light-switch1. The light1 was on. Elvis went to outside. Officer1 arrested Elvis because he wanted to control Elvis.

--- The End ---

#### Story number 7 . . .

One day ...DAD was BORED. Dad asked Karen, "Would you push the ball1 to me away from you?" Karen went to the garage. She picked up the ball1. She had the ball1. She went to outside. He went to outside. He played with the ball1. She pushed the ball1 to the calla-lilly1. He picked up the ball1. He had the ball1. He pushed the ball1 to the window1. The window1 was sharp. The window1 was shattered. She picked up the ball1. She had the ball1. H

--- The End ---

#### Story number 8 . . .

One day ...DAD was THIRSTY. Dad pushed cupboard-door away from the cupboard1. The cupboard1 was open. He took the glass2 from the cupboard1. He had the glass2. The cupboard1 didn't have the glass2. He pushed cupboard-door to the cupboard1. The cupboard1 wasn't open. He pushed cold-faucet-handle away from the cold-faucet. The cold-faucet was flowing. He moved the glass2 to the cold-faucet. The glass2 was filled with the cold-water2. He pushed cold-faucet-handle to the cold-faucet. The cold-faucet wasn't flowing. He drank the cold-water2. The glass2 wasn't filled with the cold-water2. The glass2 was dirty. He pushed hot-faucet-handle away from the hot-faucet. The hot-faucet was flowing. He moved the glass2 to the hot-faucet. The glass2 wasn't dirty. He pushed hot-faucet-handle to the hot-faucet. The hot-faucet wasn't flowing. He drank the cold-water2 because he didn't want to be thirsty.

--- The End ---

#### Story number 9 . . .

One day ...DAD was THIRSTY. Dad pushed cupboard-door away from the cupboard1. The cupboard1 was open. He took the glass4 from the cupboard1. He had the glass4. The cupboard1 didn't have the glass4. He pushed cupboard-door to the cupboard1. The cupboard1 wasn't open. He pushed fridge-door away from the fridge1. The fridge1 was open. He took the cold-water1 from the fridge1. He had the cold-water1. The fridge1 didn't have the cold-water1. He pushed fridge-door to the fridge1. The fridge1 wasn't open. He poured the cold-water1 into the glass4. The glass4 was filled with the cold-water1. He drank the cold-water1. The glass4 wasn't filled with the cold-water1. The glass4 was dirty. He pushed hot-faucet-handle away from the hot-faucet. The hot-faucet was flowing. He moved the glass4 to the hot-faucet. The glass4 wasn't dirty. Police-and-dogs arrived. Officer1 went to outside. The police-dog1 went to outside. He pushed door-bell-switch1. The door-bell1 was ringing. He didn't push door-bell-switch1. The door-bell1 wasn't ringing. He went to the kitchen. The police-dog1 went to the kitchen. The police-dog1 went to the vase2. The police-dog1 sniffed the vase2. The police-dog1 barked at the vase2. The police-dog1 was barking. He went to the vase2. He took the ganjal from the vase2. He had the ganjal. The vase2 didn't have the ganjal. He arrested Elvis. He controlled Elvis. He arrested Elvis because he wanted to control Elvis. He went to outside. Elvis went to outside. The police-dog1 went to outside. The police-dog1 barked at the vase2 because the police-dog1 detected the ganjal. Dad pushed hot-faucet-handle to the hot-faucet. The hot-faucet wasn't flowing. He drank the cold-water1 because he didn't want to be thirsty.

--- The End ---

#### Story number 10 . . .

One day ...DAD was BORED. Dad asked Karen, "Would you push the balloon2 to me away from you?" He asked him, "Would I take the balloon2 from me?" He pushed cupboard-door away from the cupboard1. The cupboard1 was open. Police-and-dogs arrived. Officer1 went to outside. The police-dog1 went to outside. He pushed door-bell-switch1. The door-bell1 was ringing. He didn't push door-bell-switch1. The door-bell1 wasn't ringing. The phone1 was ringing. Mom picked up phone-receiver1. The phone1 wasn't ringing. She had phone-receiver1. She let go of phone-receiver1. She didn't have phone-receiver1. He went to the kitchen. The police-dog1 went to the kitchen. The police-dog1 went to the rug1. The police-dog1 barked at the rug1. The police-dog1 was barking. He went to outside. The police-dog1 went to outside. He went to the kitchen. He asked Elvis whether Elvis would give him the ganjal. Elvis took the ganjal from the vase1. He had the ganjal. The vase1 didn't have the ganjal. He gave officer1 the ganjal. Officer1 had the ganjal. Elvis didn't have the ganjal. Officer1 arrested Elvis. He controlled Elvis. He went to outside. Elvis went to outside. Officer1 arrested Elvis because he wanted to control Elvis. Dad took the balloon2 from the cupboard1. He had the balloon2. The cupboard1

didn't have the balloon2. She pushed light-switch1. The light1 was on. He pushed cupboard-door to the cupboard1. The cupboard1 wasn't open. He exhaled the air2 into the balloon2. The balloon2 was inflated. He tied the balloon2. The balloon2 was sealed. He took the balloon2 from him. He had the balloon2. He didn't have the balloon2. He went to outside. Karen went to outside. He played with the balloon2. She pushed the balloon2 to the calla-lilly1. He picked up the balloon2. He had the balloon2. He pushed the balloon2 to the calla-lilly2. She picked up the balloon2. She had the balloon2. The cat1 pushed the vase2 to the floor1. The vase2 was broken. He played with the balloon2 because he didn't want to be bored.

--- The End ---

#### Story number 11 . . .

One day ...LYNN was BORED. Lynn asked Karen, "Would you push the ball2 to me away from you?" Karen went to the garage. She picked up the ball2. She had the ball2. She went to outside. Lynn went to outside. She played with the ball2. Karen hit the ball2. She hit the ball2 because she wanted to move the ball2 to Lynn. Lynn pushed the ball2 to the calla-lilly2. Karen picked up the ball2. She had the ball2. Lynn played with the ball2 because she didn't want to be bored.

--- The End ---

#### Story number 12 . . .

One day ...OFFICER1 was CONCERNED. Officer1 went to the kitchen. He asked Elvis whether Elvis would give him the ganjal. He went to outside. He couldn't get Elvis.

--- The End ---

#### Story number 13 . . .

One day ...MOM was THIRSTY. Mom pushed cupboard-door away from the cupboard1. The cupboard1 was open. She took the cup1 from the cupboard1. She had the cup1. The cupboard1 didn't have the cup1. She pushed cupboard-door to the cupboard1. The cupboard1 wasn't open. She pushed fridge-door away from the fridgel. The fridgel was open. She took the cold-water1 from the fridgel. She had the cold-water1. The fridgel didn't have the cold-water1. The police arrived. Officer1 went to

outsidel. The dog1 barked. The dog1 was barking. He pushed door-bell-switch1. The door-bell1 was ringing. He didn't push door-bell-switch1. The door-bell1 wasn't ringing. The cat1 pushed the vase2 to the floor1. The vase2 was broken. He went to the kitchen. He asked Elvis whether Elvis would give him the ganjal. Elvis took the ganjal from the vase2. He had the ganjal. The vase2 didn't have the ganjal. He gave officer1 the ganjal. Officer1 had the ganjal. Elvis didn't have the ganjal. Officer1 arrested Elvis. He controlled Elvis. He went to outside. Elvis went to outside. Officer1 arrested Elvis because he wanted to control Elvis. She pushed fridge-door to the fridgel. The fridgel wasn't open. She pushed light-switch1. The light1 was on. The phonel was ringing. Dad picked up phone-receiver1. The phonel wasn't ringing. He had phone-receiver1. He let go of phone-receiver1. He didn't have phone-receiver1. She poured the cold-water1 into the cup1. The cup1 was filled with the cold-water1. She drank the cold-water1. The cup1 wasn't filled with the cold-water1. The cup1 was dirty. She pushed hot-faucet-handle away from the hot-faucet. The hot-faucet was flowing. She moved the cup1 to the hot-faucet. The cup1 wasn't dirty. She pushed hot-faucet-handle to the hot-faucet. The hot-faucet wasn't flowing. She drank the cold-water1 because she didn't want to be thirsty.

--- The End ---

#### Story number 14 . . .

One day ...MOM was BORED. Mom asked Karen, "Would you push the ball3 to me away from you?" Karen went to the garage. She picked up the ball3. She had the ball3. She went to outside. Mom went to outside. She played with the ball3. Karen hit the ball3. She hit the ball3 because she wanted to move the ball3 to Mom. Mom hit the ball3. She hit the ball3 because she wanted to move the ball3 to Karen. Karen hit the ball3. She hit the ball3 because she wanted to move the ball3 to Mom. Mom pushed the ball3 to the grass away from her. She didn't have the ball3. She picked up the ball3. She had the ball3. She played with the ball3 because she didn't want to be bored.

--- The End ---

#### Story number 15 . . .

One day ...LYNN was JONESING. Lynn pushed cupboard-door away from the cupboard1. The cupboard1 was open. She took the pipe2 from the cupboard1. She had the pipe2. The cupboard1 didn't have the pipe2. She

pushed cupboard-door to the cupboard1. The cupboard1 wasn't open. Mom pushed light-switch1. The light1 was on. Lynn took the pipe-tobacco1 from the table1. She had the pipe-tobacco1. The table1 didn't have the pipe-tobacco1. She poured the pipe-tobacco1 into the pipe2. The pipe2 was filled with the pipe-tobacco1. She pushed the pipe2 to the floor1. The pipe2 wasn't filled with the pipe-tobacco1. She pushed cupboard-door away from the cupboard1. The cupboard1 was open. The cat1 pushed the vase2 to the floor1. The vase2 was broken. She took the pipe1 from the cupboard1. She had the pipe1. The cupboard1 didn't have the pipe1. She pushed cupboard-door to the cupboard1. The cupboard1 wasn't open. She poured the pipe-tobacco1 into the pipe1. The pipe1 was filled with the pipe-tobacco1. The phone1 was ringing. Mom picked up phone-receiver1. The phone1 wasn't ringing. She had phone-receiver1. She let go of phone-receiver1. She didn't have phone-receiver1. Lynn pushed the pipe1 to the floor1. The pipe1 wasn't filled with the pipe-tobacco1. The pipe1 was broken. She was still jonesing.

--- The End ---

#### Story number 16 . . .

One day ...DAD was JONESING. Dad pushed cupboard-door away from the cupboard1. The cupboard1 was open. He took the pipe2 from the cupboard1. He had the pipe2. The cupboard1 didn't have the pipe2. He pushed cupboard-door to the cupboard1. The cupboard1 wasn't open. Police-and-dogs arrived. Officer1 went to outside. The police-dog1 went to outside. He pushed door-bell-switch1. The door-bell1 was ringing. He didn't push door-bell-switch1. The door-bell1 wasn't ringing. He went to the kitchen. The police-dog1 went to the kitchen. The police-dog1 went to the vaset. The police-dog1 sniffed the vaset. The police-dog1 barked at the vaset. The police-dog1 was barking. He went to the vaset. He took the ganjal from the vaset. He had the ganjal. The vaset didn't have the ganjal. He arrested Elvis. He controlled Elvis. He arrested Elvis because he wanted to control Elvis. He went to outside. Elvis went to outside. The police-dog1 went to outside. The police-dog1 barked at the vaset because the police-dog1 detected the ganjal. Dad took the pipe-tobacco1 from the table1. He had the pipe-tobacco1. The table1 didn't have the pipe-tobacco1. He poured the pipe-tobacco1 into the pipe2. The pipe2 was filled with the pipe-tobacco1. He pushed the pipe2 to the floor1. The pipe2 wasn't filled with the pipe-tobacco1. He pushed cupboard-door away from the cupboard1. The cupboard1 was open. He took the pipe1 from the cupboard1. He had the pipe1. The cupboard1 didn't have the pipe1. He pushed cupboard-door to the cupboard1. The cupboard1 wasn't open. He poured the pipe-tobacco1 into the pipe1. The pipe1 was filled with the pipe-tobacco1. He pushed the pipe1 to the floor1. The pipe1 wasn't filled with the pipe-tobacco1. The pipe1 was broken. He was still jonesing.

--- The End ---

### Story number 17 . . .

One day ...ELVIS was JONESING. Elvis pushed cupboard-door away from the cupboard1. The cupboard1 was open. He took the pipel from the cupboard1. He had the pipel. The cupboard1 didn't have the pipel. He pushed cupboard-door to the cupboard1. The cupboard1 wasn't open. Mom pushed light-switch1. The light1 was on. He pushed cupboard-door away from the cupboard1. The cupboard1 was open. He took the ganjal from the cupboard1. He had the ganjal. The cupboard1 didn't have the ganjal. He pushed cupboard-door to the cupboard1. The cupboard1 wasn't open. The phonel was ringing. Dad picked up phone-receiver1. The phonel wasn't ringing. He had phone-receiver1. He let go of phone-receiver1. He didn't have phone-receiver1. Elvis poured the ganjal into the pipel. The pipel was filled with the ganjal. He took the lighter1 from the table2. He had the lighter1. The table2 didn't have the lighter1. The police arrived. Officer1 went to outside1. The dog1 barked. The dog1 was barking. He pushed door-bell-switch1. The door-bell1 was ringing. He didn't push door-bell-switch1. The door-bell1 wasn't ringing. He went to the kitchen. He asked Elvis whether Elvis would give him the ganjal. He went to the vasel. He knew, " The ganjal isn't with the vasel." He went to the rug1. The cat1 pushed the vase2 to the floor1. The vase2 was broken. He knew, " The ganjal isn't with the rug1." He went to Elvis. He knew, "The ganjal is with Elvis." He arrested Elvis. He controlled Elvis. He went to outside. Elvis went to outside. Officer1 arrested Elvis because he wanted to control Elvis. Elvis pushed the lighter1. The lighter1 was on. He had the lighter1. The ganjal couldn't be burning. Officer1 controlled Elvis. Elvis couldn't get near the pipe2. He couldn't get the pipe2. The ganjal couldn't fill the pipe2. He was still jonesing.

--- The End ---

### Story number 18 . . .

One day ...OFFICER1 was CONCERNED. Officer1 went to the kitchen. The police-dog1 went to the kitchen. The police-dog1 went to the vase3. The phonel was ringing. Dad picked up phone-receiver1. The phonel wasn't ringing. He had phone-receiver1. He let go of phone-receiver1. He didn't have phone-receiver1. Mom pushed light-switch1. The light1 was on. The police-dog1 sniffed the vase3. The police-dog1 barked at the vase3. The police-dog1 was barking. Officer1 went to the vase3. He took the ganjal from the vase3. He had the ganjal. The vase3 didn't have the ganjal. He arrested Elvis. He controlled Elvis. He arrested Elvis because he wanted to control Elvis. He went to outside. Elvis went to outside. The police-dog1 went to outside. The police-dog1 barked at the vase3 because the

police-dog1 detected the ganjal.

--- The End ---

### Story number 19 . . .

One day ...DAD was JONESING. Dad pushed cupboard-door away from the cupboard1. The cupboard1 was open. He took the pipe1 from the cupboard1. He had the pipe1. The cupboard1 didn't have the pipe1. He pushed cupboard-door to the cupboard1. The cupboard1 wasn't open. He took the pipe-tobacco1 from the table1. He had the pipe-tobacco1. The table1 didn't have the pipe-tobacco1. He poured the pipe-tobacco1 into the pipe1. The pipe1 was filled with the pipe-tobacco1. He pushed the pipe1 to the floor1. The pipe1 wasn't filled with the pipe-tobacco1. The pipe1 was broken. He pushed cupboard-door away from the cupboard1. The cupboard1 was open. He took the pipe2 from the cupboard1. He had the pipe2. The cupboard1 didn't have the pipe2. He pushed cupboard-door to the cupboard1. The cupboard1 wasn't open. He poured the pipe-tobacco1 into the pipe2. The pipe2 was filled with the pipe-tobacco1. He pushed the pipe2 to the floor1. The pipe2 wasn't filled with the pipe-tobacco1. He was still jonesing.

--- The End ---

### Story number 20 . . .

One day ...DAD was BORED. Dad asked Karen, "Would you push the balloon3 to me away from you?" He asked Mom, "Would you give me the balloon3?" Mom pushed cupboard-door away from the cupboard1. The cupboard1 was open. She took the balloon3 from the cupboard1. She had the balloon3. The cupboard1 didn't have the balloon3. She pushed cupboard-door to the cupboard1. The cupboard1 wasn't open. She exhaled the air2 into the balloon3. The balloon3 was inflated. She let go of the balloon3. She didn't have the balloon3. The balloon3 was flying. The balloon3 wasn't inflated. She picked up the balloon3. She had the balloon3. She exhaled the air2 into the balloon3. The balloon3 was inflated. She let go of the balloon3. She didn't have the balloon3. The balloon3 was flying. The balloon3 wasn't inflated. She picked up the balloon3. She had the balloon3. She exhaled the air2 into the balloon3. The balloon3 was inflated. She tied the balloon3. The balloon3 was sealed. She gave him the balloon3. He had the balloon3. She didn't have the balloon3. He went to outside. Karen went to outside. He played with the balloon3. She pushed the balloon3 to him away from her. He had the balloon3. She didn't have the balloon3. The police arrived. Officer1 went to outside1. The dog1 barked. The dog1 was barking. He pushed door-bell-switch1. The door-bell1 was ringing. He didn't push door-bell-switch1. The door-bell1 wasn't ringing. He went to the kitchen. He asked Elvis whether Elvis would give him the ganjal. The phone1 was ringing. Mom picked up phone-



receiver1. The phone1 wasn't ringing. She had phone-receiver1. She let go of phone-receiver1. She didn't have phone-receiver1. The cat1 pushed the vase2 to the floor1. The vase2 was broken. She pushed light-switch1. The light1 was on. Elvis took the ganjal from the rug1. He had the ganjal. The rug1 didn't have the ganjal. He gave officer1 the ganjal. Officer1 had the ganjal. Elvis didn't have the ganjal. Officer1 went to outside. He couldn't get Elvis. Dad pushed the balloon3 to Karen away from him. Karen had the balloon3. He didn't have the balloon3. She pushed the balloon3 to him away from her. He had the balloon3. She didn't have the balloon3. He played with the balloon3 because he didn't want to be bored.

--- The End ---

#### Story number 21 . . .

One day ...LYNN was JONESING. Lynn pushed cupboard-door away from the cupboard1. The cupboard1 was open. She took the pipe2 from the cupboard1. She had the pipe2. The cupboard1 didn't have the pipe2. She pushed cupboard-door to the cupboard1. The cupboard1 wasn't open. She took the pipe-tobacco1 from the table1. She had the pipe-tobacco1. The table1 didn't have the pipe-tobacco1. The police arrived. Officer1 went to outside1. The dog1 barked. The dog1 was barking. He pushed door-bell-switch1. The door-bell1 was ringing. He didn't push door-bell-switch1. The door-bell1 wasn't ringing. He went to the kitchen. He asked Elvis whether Elvis would give him the ganjal. Elvis took the ganjal from the rug1. He had the ganjal. The rug1 didn't have the ganjal. He gave officer1 the ganjal. Officer1 had the ganjal. Elvis didn't have the ganjal. Officer1 went to outside. He couldn't get Elvis. She poured the pipe-tobacco1 into the pipe2. The pipe2 was filled with the pipe-tobacco1. She took the lighter1 from the table2. She had the lighter1. The table2 didn't have the lighter1. She pushed the lighter1. The lighter1 was on. She moved the lighter1 to the pipe-tobacco1. The pipe-tobacco1 was burning. She pushed the lighter1. The lighter1 wasn't on. She smoked the pipe-tobacco1. The pipe2 wasn't filled with the pipe-tobacco1. The pipe2 was dirty. She exhaled the smoke1 into the air1. She pushed hot-faucet-handle away from the hot-faucet. The hot-faucet was flowing. She moved the pipe2 to the hot-faucet. The pipe2 wasn't dirty. She pushed hot-faucet-handle to the hot-faucet. The hot-faucet wasn't flowing. She smoked the pipe-tobacco1 because she didn't want to be withdrawing.

--- The End ---

#### Story number 22 . . .

One day ...DAD was BORED. Dad asked Karen, "Would you push the ball2 to me away from you?" Karen went to the garage. She picked up the ball2. She had the ball2. She went to outside. He went to outside. He played with the ball2. She pushed the ball2 to the window1. The window1 was

sharp. The window1 was shattered. He picked up the ball2. He had the ball2. He pushed the ball2 to the rose-bush1. She picked up the ball2. She had the ball2. The cat1 pushed the vase2 to the floor1. The vase2 was broken. She hit the ball2. She hit the ball2 because she wanted to move the ball2 to him. He played with the ball2 because he didn't want to be bored.

--- The End ---

### Story number 23 . . .

One day ...ELVIS was THIRSTY. Elvis pushed cupboard-door away from the cupboard1. The cupboard1 was open. He took the glass1 from the cupboard1. He had the glass1. The cupboard1 didn't have the glass1. He pushed cupboard-door to the cupboard1. The cupboard1 wasn't open. He pushed cold-faucet-handle away from the cold-faucet. The cold-faucet was flowing. He moved the glass1 to the cold-faucet. The glass1 was filled with the cold-water2. He pushed cold-faucet-handle to the cold-faucet. The cold-faucet wasn't flowing. He drank the cold-water2. The glass1 wasn't filled with the cold-water2. The glass1 was dirty. Mom pushed light-switch1. The light1 was on. Police-and-dogs arrived. Officer1 went to outside. The police-dog1 went to outside. He pushed door-bell-switch1. The door-bell1 was ringing. He didn't push door-bell-switch1. The door-bell1 wasn't ringing. He went to the kitchen. The police-dog1 went to the kitchen. The police-dog1 went to the laundry-pile1. The police-dog1 sniffed the laundry-pile1. The police-dog1 barked at the laundry-pile1. The police-dog1 was barking. He went to the laundry-pile1. He took the ganjal from the laundry-pile1. He had the ganjal. The laundry-pile1 didn't have the ganjal. He arrested Elvis. He controlled Elvis. He arrested Elvis because he wanted to control Elvis. He went to outside. Elvis went to outside. The police-dog1 went to outside. The police-dog1 barked at the laundry-pile1 because the police-dog1 detected the ganjal. He pushed hot-faucet-handle away from the hot-faucet. The hot-faucet was flowing. He had the glass1.

--- The End ---

### Story number 24 . . .

One day ...MOM was BORED. Mom asked Karen, "Would you push the ball3 to me away from you?" Karen went to the garage. She picked up the ball3. She had the ball3. The police arrived. Officer1 went to outside1. The dog1 barked. The dog1 was barking. He pushed door-bell-switch1. The

door-bell1 was ringing. He didn't push door-bell-switch1. The door-bell1 wasn't ringing. The phone1 was ringing. Dad picked up phone-receiver1. The phone1 wasn't ringing. He had phone-receiver1. He let go of phone-receiver1. He didn't have phone-receiver1. Officer1 went to the kitchen. The cat1 pushed the vase2 to the floor1. The vase2 was broken. He asked Elvis whether Elvis would give him the ganjal. Elvis pushed cupboard-door away from the cupboard1. The cupboard1 was open. He took the ganjal from the cupboard1. He had the ganjal. The cupboard1 didn't have the ganjal. He pushed cupboard-door to the cupboard1. The cupboard1 wasn't open. He gave officer1 the ganjal. Officer1 had the ganjal. Elvis didn't have the ganjal. Officer1 went to outside. Mom pushed light-switch1. The light1 was on. He couldn't get Elvis. Karen went to outside. Mom went to outside. She played with the ball3. Karen pushed the ball3 to the window1. The window1 was sharp. The window1 was shattered. Mom picked up the ball3. She had the ball3. She pushed the ball3 to the rose-bush2. Karen picked up the ball3. She had the ball3. Mom played with the ball3 because she didn't want to be bored.

--- The End ---



## APPENDIX D

### META-AQUA OUTPUT IN LISP PROGRAMMING MODE

Begin Meta-AQUA.

Continue ? (Y or N) Yes.

Input Structure: EACH-ONE-GREATER.241

NIL

Continue ? (Y or N) Yes.

Input Structure: EACH-ONE-GREATER.241

NIL

Continue ? (Y or N) Yes.

Retrieving plan for concept EACH-ONE-GREATER.241

Produced plan: RECURSIVE-PROGRAMMING-PLAN.462.

Continue ? (Y or N) Yes.

Actor PERSON.0 performs act WRITE-DEFUN.229

and expects result NIL

Continue ? (Y or N) Yes.

Input Structure: WRITE-DEFUN.229

NIL

Continue ? (Y or N) Yes.

WRITE-DEFUN.229 is not a very interesting concept.

Skimming . . .

Continue ? (Y or N) Yes.  
Actor PERSON.0 performs act WRITE-NAME.230

and expects result NIL

Continue ? (Y or N) Yes.  
Input Structure: WRITE-NAME.230

NIL

Continue ? (Y or N) Yes.  
WRITE-NAME.230 is not a very interesting concept.

Skimming . . .

Continue ? (Y or N) Yes.  
Actor PERSON.0 performs act ADD-PARAMETERS.231

and expects result NIL

Continue ? (Y or N) Yes.  
Input Structure: ADD-PARAMETERS.231

NIL

Continue ? (Y or N) Yes.  
ADD-PARAMETERS.231 is not a very interesting concept.

Skimming . . .

Continue ? (Y or N) Yes.  
Actor PERSON.0 performs act WRITE-BODY.232  
  
and expects result NEWLY-LEARNED-ADD1NUMS-DEFUN.228

LISP simulator provides alternative solution  
RECURSIVE-ADD1NUMS-DEFUN.1110

Continue ? (Y or N) Yes.  
Input Structure: WRITE-BODY.232

NIL

Continue ? (Y or N) Yes.  
 ;Now the following should actually be interesting to the program.  
 WRITE-BODY.232 is not a very interesting concept.

Skimming . . .

Continue ? (Y or N) Yes.

Continue ? (Y or N) Yes.  
 Input Structure: EACH-ONE-GREATER.226

NIL

Continue ? (Y or N) Yes.  
 EACH-ONE-GREATER.226 is not a very interesting concept.

Skimming . . .

Continue ? (Y or N) No.  
 Done.  
 LISP-PROGRAMMING  
 Command:

;The following would be the result of the rough comparison.  
 ;It creates the expectation-failure knowledge structure.  
 Command: (compare 'NEWLY-LEARNED-ADD1NUMS-DEFUN.2280  
                   'RECURSIVE-ADD1NUMS-DEFUN.1110  
                   'goal.524)

MENTALLY-INITIATES.1632

;Pretty print it two levels deep.  
 Command: (f.pprint 'mentally-initiates.1632 2)

Frame MENTALLY-INITIATES.1632 has the following value:

```
(INITIATES
  (DOMAIN NOT-EQUAL-RELATION.1633)
  (CO-DOMAIN EXPECTATION-FAILURE.1634)
)
```

Frame NOT-EQUAL-RELATION.1633 has the following value:

```
(RELATION
  (DOMAIN NEWLY-LEARNED-ADD1NUMS-DEFUN.228)
  (CO-DOMAIN RECURSIVE-ADD1NUMS-DEFUN.1110)
)
```

Frame EXPECTATION-FAILURE.1634 has the following value:

```
(COMMISSION-ERROR
  (INITIATES- NOT-EQUAL-RELATION.1633)
  (EXPECTED-OUTCOME RECURSIVE-ADD1NUMS-DEFUN.1110)
  (ACTUAL-OUTCOME NEWLY-LEARNED-ADD1NUMS-DEFUN.228)
)
```

NIL

```
;The following would represent the fine level comparison
;from the reflect on error strategy. Note that it actually
;localizes the specific difference that caused the failure:
;The subject did not make a recursive call to the add1nums function.
Command: (f.unify 'NEWLY-LEARNED-ADD1NUMS-DEFUN.228
                  'RECURSIVE-ADD1NUMS-DEFUN.1110)
```

Unification fails.

Attempt to merge LIST-OP.501 and RECURSIVE-ADD1NUMS-DEFUN.1121

```
along path (BODY SCENES ACTION-CLAUSES OLD-LIST)
```

```
during unify call on NEWLY-LEARNED-ADD1NUMS-DEFUN.228
and RECURSIVE-ADD1NUMS-DEFUN.1110.
```

Do you wish to BREAK ? (Y or N) No.

NIL

```
(LIST-OP.501 RECURSIVE-ADD1NUMS-DEFUN.1121)
(BODY SCENES ACTION-CLAUSES OLD-LIST)
```

Command:

```
;The following is a trace of goals in the priority queue.
Goal Queue --> (GOAL.245)
Next Goal   --> GOAL.245
```

Frame GOAL.245 has the following value:

```
(MENTAL-STATE
  (GOAL-ACTOR PERSON.0)
```



```

(GOAL-OBJECT WANTS.244)
(GOAL-TYPE ACHIEVEMENT-GOAL.0)
(PRIORITY SIX.0)
(ACHIEVED FALSE.0)
(DOMAIN PERSON.0)
(CO-DOMAIN WANTS.244)
(SUPERGOAL NIL.0)
(GOAL-VALUE AMOUNT-VALUE.256)
(BACKPTR PLAN.257)
(MXP TRACE-META-XP.258)
)

Goal-State --> PERSON.0 WANTS EACH-ONE-GRE
ATER.241
-----

Goal Queue --> (GOAL.319 GOAL.245)
Next Goal   --> GOAL.319

```

Frame GOAL.319 has the following value:

```

(MENTAL-STATE
  (GOAL-ACTOR PERSON.0)
  (GOAL-OBJECT ID.318)
  (GOAL-TYPE ACHIEVEMENT-GOAL.320)
  (PRIORITY SEVEN.0)
  (ACHIEVED FALSE.0)
  (DOMAIN PERSON.0)
  (CO-DOMAIN ID.318)
  (SUPERGOAL NIL.0)
  (GOAL-VALUE AMOUNT-VALUE.332)
  (BACKPTR PLAN.333)
  (MXP TRACE-META-XP.334)
)

Goal-State --> PERSON.0 ID EACH-ONE-GREATE
R.241
-----

Goal Queue --> (GOAL.402 GOAL.319
                GOAL.245)
Next Goal   --> GOAL.402

```

Frame GOAL.402 has the following value:

```

(MENTAL-STATE
  (GOAL-ACTOR PERSON.0)
  (GOAL-OBJECT GENERATE.401)
  (GOAL-TYPE ACHIEVEMENT-GOAL.403)
)

```

```

(PRIORITY SEVEN.0)
(ACHIEVED FALSE.0)
(DOMAIN PERSON.0)
(CO-DOMAIN GENERATE.401)
(SUPERGOAL GOAL.319)
(GOAL-VALUE AMOUNT-VALUE.415)
(BACKPTR PLAN.416)
(MXP TRACE-META-XP.417)
)

Goal-State --> PERSON.0 GENERATE EACH-ONE-
GREATER.241
-----

```

```

Goal Queue --> (GOAL.524 GOAL.402
                GOAL.319 GOAL.245)
Next Goal   --> GOAL.524

```

Frame GOAL.524 has the following value:

```

(MENTAL-STATE
  (GOAL-ACTOR PERSON.0)
  (GOAL-OBJECT TEST.523)
  (GOAL-TYPE ACHIEVEMENT-GOAL.525)
  (PRIORITY SEVEN.0)
  (ACHIEVED FALSE.0)
  (DOMAIN PERSON.0)
  (CO-DOMAIN TEST.523)
  (SUPERGOAL GOAL.402)
  (GOAL-VALUE AMOUNT-VALUE.537)
  (BACKPTR PLAN.538)
  (MXP TRACE-META-XP.539)
)

```

```

Goal-State --> PERSON.0 TEST RECURSIVE-PRO
GRAMMING-PLAN.462
-----

```

```

Goal Queue --> (GOAL.582 GOAL.524
                GOAL.402 GOAL.319
                GOAL.245)
Next Goal   --> GOAL.582

```

Frame GOAL.582 has the following value:

```

(MENTAL-STATE
  (GOAL-ACTOR PERSON.0)
  (GOAL-OBJECT UNDERSTANDS.581)
  (GOAL-TYPE KNOWLEDGE-ACQUISITION-GOAL.58

```

```

3)
  (PRIORITY SEVEN.0)
  (ACHIEVED FALSE.0)
  (DOMAIN PERSON.0)
  (CO-DOMAIN UNDERSTANDS.581)
  (SUPERGOAL NIL.0)
  (GOAL-VALUE AMOUNT-VALUE.595)
  (BACKPTR RECURSIVE-PROGRAMMING-PLAN.462)
  (MXP NIL.0)
)

```

```

Goal-State --> PERSON.0 UNDERSTANDS WRITE-
DEFUN.229
-----

```

```

Goal Queue --> (GOAL.658 GOAL.582
                GOAL.524 GOAL.402
                GOAL.319 GOAL.245)
Next Goal   --> GOAL.658

```

Frame GOAL.658 has the following value:

```

(MENTAL-STATE
  (GOAL-ACTOR PERSON.0)
  (GOAL-OBJECT ID.657)
  (GOAL-TYPE KNOWLEDGE-ACQUISITION-GOAL.65
9)
  (PRIORITY SEVEN.0)
  (ACHIEVED FALSE.0)
  (DOMAIN PERSON.0)
  (CO-DOMAIN ID.657)
  (SUPERGOAL GOAL.582)
  (GOAL-VALUE AMOUNT-VALUE.671)
  (BACKPTR PLAN.672)
  (MXP TRACE-META-XP.673)
)

```

```

Goal-State --> PERSON.0 ID WRITE-DEFUN.229
-----

```

```

Goal Queue --> (GOAL.524 GOAL.402
                GOAL.319 GOAL.245)
Next Goal   --> GOAL.524

```

Frame GOAL.524 has the following value:

```

(MENTAL-STATE
  (GOAL-ACTOR PERSON.0)
  (GOAL-OBJECT TEST.523)
)

```

```

    (GOAL-TYPE ACHIEVEMENT-GOAL.525)
    (PRIORITY SEVEN.0)
    (ACHIEVED FALSE.0)
    (DOMAIN PERSON.0)
    (CO-DOMAIN TEST.523)
    (SUPERGOAL GOAL.402)
    (GOAL-VALUE AMOUNT-VALUE.537)
    (BACKPTR PLAN.538)
    (MXP TRACE-META-XP.539)
)

```

```

Goal-State --> PERSON.0 TEST RECURSIVE-PRO
GRAMMING-PLAN.462
-----

```

```

Goal Queue --> (GOAL.759 GOAL.524
                GOAL.402 GOAL.319
                GOAL.245)
Next Goal   --> GOAL.759

```

Frame GOAL.759 has the following value:

```

(MENTAL-STATE
  (GOAL-ACTOR PERSON.0)
  (GOAL-OBJECT UNDERSTANDS.758)
  (GOAL-TYPE KNOWLEDGE-ACQUISITION-GOAL.76
0)
  (PRIORITY SEVEN.0)
  (ACHIEVED FALSE.0)
  (DOMAIN PERSON.0)
  (CO-DOMAIN UNDERSTANDS.758)
  (SUPERGOAL NIL.0)
  (GOAL-VALUE AMOUNT-VALUE.772)
  (BACKPTR RECURSIVE-PROGRAMMING-PLAN.462)
  (MXP NIL.0)
)

```

```

Goal-State --> PERSON.0 UNDERSTANDS WRITE-
NAME.230
-----

```

```

Goal Queue --> (GOAL.834 GOAL.759
                GOAL.524 GOAL.402
                GOAL.319 GOAL.245)
Next Goal   --> GOAL.834

```

Frame GOAL.834 has the following value:

```

(MENTAL-STATE

```

```

      (GOAL-ACTOR PERSON.0)
      (GOAL-OBJECT ID.833)
      (GOAL-TYPE KNOWLEDGE-ACQUISITION-GOAL.83
5)
      (PRIORITY SEVEN.0)
      (ACHIEVED FALSE.0)
      (DOMAIN PERSON.0)
      (CO-DOMAIN ID.833)
      (SUPERGOAL GOAL.759)
      (GOAL-VALUE AMOUNT-VALUE.847)
      (BACKPTR PLAN.848)
      (MXP TRACE-META-XP.849)
)

```

```
Goal-State --> PERSON.0 ID WRITE-NAME.230
-----
```

```
Goal Queue --> (GOAL.524 GOAL.402
                GOAL.319 GOAL.245)
Next Goal   --> GOAL.524

```

Frame GOAL.524 has the following value:

```

(MENTAL-STATE
  (GOAL-ACTOR PERSON.0)
  (GOAL-OBJECT TEST.523)
  (GOAL-TYPE ACHIEVEMENT-GOAL.525)
  (PRIORITY SEVEN.0)
  (ACHIEVED FALSE.0)
  (DOMAIN PERSON.0)
  (CO-DOMAIN TEST.523)
  (SUPERGOAL GOAL.402)
  (GOAL-VALUE AMOUNT-VALUE.537)
  (BACKPTR PLAN.538)
  (MXP TRACE-META-XP.539)
)

```

```
Goal-State --> PERSON.0 TEST RECURSIVE-PRO
GRAMMING-PLAN.462
-----
```

```
Goal Queue --> (GOAL.935 GOAL.524
                GOAL.402 GOAL.319
                GOAL.245)
Next Goal   --> GOAL.935

```

Frame GOAL.935 has the following value:

```
(MENTAL-STATE
```

```

        (GOAL-ACTOR PERSON.0)
        (GOAL-OBJECT UNDERSTANDS.934)
        (GOAL-TYPE KNOWLEDGE-ACQUISITION-GOAL.93
6)
        (PRIORITY SEVEN.0)
        (ACHIEVED FALSE.0)
        (DOMAIN PERSON.0)
        (CO-DOMAIN UNDERSTANDS.934)
        (SUPERGOAL NIL.0)
        (GOAL-VALUE AMOUNT-VALUE.948)
        (BACKPTR RECURSIVE-PROGRAMMING-PLAN.462)
        (MXP NIL.0)
)

```

```

Goal-State --> PERSON.0 UNDERSTANDS ADD-PARAMETERS.231
-----

```

```

Goal Queue --> (GOAL.1010 GOAL.935
                  GOAL.524 GOAL.402
                  GOAL.319 GOAL.245)
Next Goal   --> GOAL.1010

```

Frame GOAL.1010 has the following value:

```

(MENTAL-STATE
  (GOAL-ACTOR PERSON.0)
  (GOAL-OBJECT ID.1009)
  (GOAL-TYPE KNOWLEDGE-ACQUISITION-GOAL.10
11)
  (PRIORITY SEVEN.0)
  (ACHIEVED FALSE.0)
  (DOMAIN PERSON.0)
  (CO-DOMAIN ID.1009)
  (SUPERGOAL GOAL.935)
  (GOAL-VALUE AMOUNT-VALUE.1023)
  (BACKPTR PLAN.1024)
  (MXP TRACE-META-XP.1025)
)

```

```

Goal-State --> PERSON.0 ID ADD-PARAMETERS.
231
-----

```

```

Goal Queue --> (GOAL.524 GOAL.402
                  GOAL.319 GOAL.245)
Next Goal   --> GOAL.524

```

Frame GOAL.524 has the following value:

```

(MENTAL-STATE
  (GOAL-ACTOR PERSON.0)
  (GOAL-OBJECT TEST.523)
  (GOAL-TYPE ACHIEVEMENT-GOAL.525)
  (PRIORITY SEVEN.0)
  (ACHIEVED FALSE.0)
  (DOMAIN PERSON.0)
  (CO-DOMAIN TEST.523)
  (SUPERGOAL GOAL.402)
  (GOAL-VALUE AMOUNT-VALUE.537)
  (BACKPTR PLAN.538)
  (MXP TRACE-META-XP.539)
)

```

```

Goal-State --> PERSON.0 TEST RECURSIVE-PRO
GRAMMING-PLAN.462
-----

```

```

Goal Queue --> (GOAL.1124 GOAL.524
                GOAL.402 GOAL.319
                GOAL.245)
Next Goal   --> GOAL.1124

```

Frame GOAL.1124 has the following value:

```

(MENTAL-STATE
  (GOAL-ACTOR PERSON.0)
  (GOAL-OBJECT UNDERSTANDS.1123)
  (GOAL-TYPE KNOWLEDGE-ACQUISITION-GOAL.11
25)
  (PRIORITY SEVEN.0)
  (ACHIEVED FALSE.0)
  (DOMAIN PERSON.0)
  (CO-DOMAIN UNDERSTANDS.1123)
  (SUPERGOAL NIL.0)
  (GOAL-VALUE AMOUNT-VALUE.1137)
  (BACKPTR RECURSIVE-PROGRAMMING-PLAN.462)
  (MXP NIL.0)
)

```

```

Goal-State --> PERSON.0 UNDERSTANDS WRITE-
BODY.232
-----

```

```

Goal Queue --> (GOAL.1200 GOAL.1124
                GOAL.524 GOAL.402
                GOAL.319 GOAL.245)
Next Goal   --> GOAL.1200

```

Frame GOAL.1200 has the following value:

```
(MENTAL-STATE
  (GOAL-ACTOR PERSON.0)
  (GOAL-OBJECT ID.1199)
  (GOAL-TYPE KNOWLEDGE-ACQUISITION-GOAL.12
01)
  (PRIORITY SEVEN.0)
  (ACHIEVED FALSE.0)
  (DOMAIN PERSON.0)
  (CO-DOMAIN ID.1199)
  (SUPERGOAL GOAL.1124)
  (GOAL-VALUE AMOUNT-VALUE.1213)
  (BACKPTR PLAN.1214)
  (MXP TRACE-META-XP.1215)
)
```

Goal-State --> PERSON.0 ID WRITE-BODY.232

```
-----
Goal Queue --> (GOAL.524 GOAL.402
                  GOAL.319 GOAL.245)
Next Goal   --> GOAL.524
```

Frame GOAL.524 has the following value:

```
(MENTAL-STATE
  (GOAL-ACTOR PERSON.0)
  (GOAL-OBJECT TEST.523)
  (GOAL-TYPE ACHIEVEMENT-GOAL.525)
  (PRIORITY SEVEN.0)
  (ACHIEVED FALSE.0)
  (DOMAIN PERSON.0)
  (CO-DOMAIN TEST.523)
  (SUPERGOAL GOAL.402)
  (GOAL-VALUE AMOUNT-VALUE.537)
  (BACKPTR PLAN.538)
  (MXP TRACE-META-XP.539)
)
```

Goal-State --> PERSON.0 TEST RECURSIVE-PRO  
GRAMMING-PLAN.462

```
-----
Goal Queue --> (GOAL.1302 GOAL.524
                  GOAL.402 GOAL.319
                  GOAL.245)
Next Goal   --> GOAL.1302
```



Frame GOAL.1302 has the following value:

```
(MENTAL-STATE
  (GOAL-ACTOR PERSON.0)
  (GOAL-OBJECT UNDERSTANDS.1301)
  (GOAL-TYPE KNOWLEDGE-ACQUISITION-GOAL.1303)
  (PRIORITY SEVEN.0)
  (ACHIEVED FALSE.0)
  (DOMAIN PERSON.0)
  (CO-DOMAIN UNDERSTANDS.1301)
  (SUPERGOAL NIL.0)
  (GOAL-VALUE AMOUNT-VALUE.1315)
  (BACKPTR RECURSIVE-PROGRAMMING-PLAN.462)
  (MXP NIL.0)
)
```

Goal-State --> PERSON.0 UNDERSTANDS EACH-ONE-GREATER.226

---

Goal Queue --> (GOAL.1376 GOAL.1302  
                   GOAL.524 GOAL.402  
                   GOAL.319 GOAL.245)

Next Goal --> GOAL.1376

Frame GOAL.1376 has the following value:

```
(MENTAL-STATE
  (GOAL-ACTOR PERSON.0)
  (GOAL-OBJECT ID.1375)
  (GOAL-TYPE KNOWLEDGE-ACQUISITION-GOAL.1377)
  (PRIORITY SEVEN.0)
  (ACHIEVED FALSE.0)
  (DOMAIN PERSON.0)
  (CO-DOMAIN ID.1375)
  (SUPERGOAL GOAL.1302)
  (GOAL-VALUE AMOUNT-VALUE.1389)
  (BACKPTR PLAN.1390)
  (MXP TRACE-META-XP.1391)
)
```

Goal-State --> PERSON.0 ID EACH-ONE-GREATER.226

---

Goal Queue --> (GOAL.524 GOAL.402  
                   GOAL.319 GOAL.245)

Next Goal --> GOAL.524

Frame GOAL.524 has the following value:

```
(MENTAL-STATE
  (GOAL-ACTOR PERSON.0)
  (GOAL-OBJECT TEST.523)
  (GOAL-TYPE ACHIEVEMENT-GOAL.525)
  (PRIORITY SEVEN.0)
  (ACHIEVED FALSE.0)
  (DOMAIN PERSON.0)
  (CO-DOMAIN TEST.523)
  (SUPERGOAL GOAL.402)
  (GOAL-VALUE AMOUNT-VALUE.537)
  (BACKPTR PLAN.538)
  (MXP TRACE-META-XP.539)
)
```

Goal-State --> PERSON.0 TEST RECURSIVE-PROGRAMMING-PLAN.462

-----

Goal Queue --> (GOAL.1479 GOAL.524  
                  GOAL.402 GOAL.319  
                  GOAL.245)

Next Goal --> GOAL.1479

Frame GOAL.1479 has the following value:

```
(MENTAL-STATE
  (GOAL-ACTOR PERSON.0)
  (GOAL-OBJECT UNDERSTANDS.1478)
  (GOAL-TYPE KNOWLEDGE-ACQUISITION-GOAL.1480)
  (PRIORITY SEVEN.0)
  (ACHIEVED FALSE.0)
  (DOMAIN PERSON.0)
  (CO-DOMAIN UNDERSTANDS.1478)
  (SUPERGOAL NIL.0)
  (GOAL-VALUE AMOUNT-VALUE.1492)
  (BACKPTR RECURSIVE-PROGRAMMING-PLAN.462)
  (MXP NIL.0)
)
```

Goal-State --> PERSON.0 UNDERSTANDS EACH-ONE-GREATER.226

-----

Goal Queue --> (GOAL.1553 GOAL.1479

```

                GOAL.524 GOAL.402
                GOAL.319 GOAL.245)
Next Goal    --> GOAL.1553

```

Frame GOAL.1553 has the following value:

```

(MENTAL-STATE
  (GOAL-ACTOR PERSON.0)
  (GOAL-OBJECT ID.1552)
  (GOAL-TYPE KNOWLEDGE-ACQUISITION-GOAL.15
54)
  (PRIORITY SEVEN.0)
  (ACHIEVED FALSE.0)
  (DOMAIN PERSON.0)
  (CO-DOMAIN ID.1552)
  (SUPERGOAL GOAL.1479)
  (GOAL-VALUE AMOUNT-VALUE.1566)
  (BACKPTR PLAN.1567)
  (MXP TRACE-META-XP.1568)
)

```

```

Goal-State  --> PERSON.0 ID EACH-ONE-GREATE
R.226
-----

```

```

Goal Queue  --> (GOAL.524 GOAL.402
                GOAL.319 GOAL.245)
Next Goal    --> GOAL.524

```

Frame GOAL.524 has the following value:

```

(MENTAL-STATE
  (GOAL-ACTOR PERSON.0)
  (GOAL-OBJECT TEST.523)
  (GOAL-TYPE ACHIEVEMENT-GOAL.525)
  (PRIORITY SEVEN.0)
  (ACHIEVED FALSE.0)
  (DOMAIN PERSON.0)
  (CO-DOMAIN TEST.523)
  (SUPERGOAL GOAL.402)
  (GOAL-VALUE AMOUNT-VALUE.537)
  (BACKPTR PLAN.538)
  (MXP TRACE-META-XP.539)
)

```

```

Goal-State  --> PERSON.0 TEST RECURSIVE-PRO
GRAMMING-PLAN.462
-----

```

## WORLD MODEL

```
-> WRITE-DEFUN.229
-> WRITE-NAME.230
-> ADD-PARAMETERS.231
-> WRITE-BODY.232
-> NEWLY-LEARNED-ADD1NUMS-DEFUN.228
-> RECURSIVE-ADD1NUMS-DEFUN.1110
-> EACH-ONE-GREATER.226
```

## REASONING MODEL

```
-> TRACE-META-XP.258
-> TRACE-META-XP.539
-> NIL.0
-> NIL.0
-> NIL.0
-> NIL.0
-> NIL.0
-> NIL.0
```

Print IMXP ? (Y or N) No.

Command: (f.pprint 'goal.524)

Frame GOAL.524 has the following value:

```
(MENTAL-STATE
  (GOAL-ACTOR PERSON.0)
  (GOAL-OBJECT TEST.523)
  (GOAL-TYPE ACHIEVEMENT-GOAL.525)
  (PRIORITY SEVEN.0)
  (ACHIEVED FALSE.0)
  (DOMAIN PERSON.0)
  (CO-DOMAIN TEST.523)
  (SUPERGOAL GOAL.402)
  (GOAL-VALUE AMOUNT-VALUE.537)
  (BACKPTR PLAN.538)
  (MXP TRACE-META-XP.539)
)
```

NIL

Command:

## REFERENCES

- Aamodt, A. (1994). Explanation-driven case-based reasoning. In S. Wess, K.-D. Althoff, and M. M. Richter (Eds.), *Topics in case-based reasoning (EWCBR-93)* (pp. 274-288). Berlin: Springer-Verlag.
- Aho, A. V., & Ullman, J. D. (1992). *Foundations of computer science*. New York: Computer Science Press.
- Allen, J., Hendler, J., & Tate, A. (Eds.). (1990). *Readings in planning*. San Mateo, CA: Morgan Kaufmann.
- Almonayyes, A. (1994). Improving problem understanding through case-based reasoning: A case study in the domain of international conflicts. In J. W. Brahan & G. E. Lasker (Eds.), *Proceedings of the Seventh International Conference on Systems Research, Informatics and Cybernetics: Vol. 2. Advances in Artificial Intelligence - Theory and Application II* (pp. 194-202). Windsor, Ontario, Canada: The International Institute for Advanced Studies in Systems Research and Cybernetics.
- Anderson, J. R. (1983). *The architecture of cognition*. Cambridge, MA: Harvard University Press.
- Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Anderson, J. R., & Reiser, B. J. (1985, April). The LISP tutor. *BYTE*, pp. 159-175.
- Anderson, J. R., & Thompson, R. (1989). Use of analogy in a production system architecture. In S. Vosniadou & A. Ortony (Eds.), *Similarity and analogical reasoning* (pp. 267-297). Cambridge: Cambridge University Press.
- Antaki, C., & Lewis, A. (Eds.) (1986). *Mental mirrors: Metacognition in social knowledge and communication*. London: Sage Publications.
- Arkin, R. (1987). *Towards cosmopolitan robots: Intelligent navigation in extended man-made environments* (Tech. Rep. No. 87-80). Doctoral dissertation, University of Massachusetts, Computer and Information Science, Amherst.
- Ashley, K. D., & McLaren, B. M. (1995). A CBR knowledge representation for practical ethics. In J.-P. Haton, M. Keane, & M. Manago (Eds.), *Advances in case-based reasoning* (pp. 181-197). Berlin: Springer-Verlag.

Ashley, K. D., & Rissland, E. L. (1988). A case-based approach to modeling legal expertise. *IEEE Expert*, 3(3), 70-77.

Asimov, I. (1942). Runaround. *Astounding Science Fiction*.

Augustine (1955). De Trinitate. In J. Burnaby (Trans. and Ed.), *Augustine: Later works* (Vol. 8, Library of Christian Classics, Bk. 10, Sec. 7, p. 80). SCM Press. (Original work published around 1600)

Ayer, A. J. (1952). *Language, truth and logic*. New York: Dover Publications. (Original work published 1936)

Bain, W. M. (1986). *Case-based reasoning: A computer model of subjective assessment*. Doctoral dissertation, Yale University, Department of Computer Science, New Haven, CT.

Ballim, A. (1993). Macro and micro attributions of mental attitudes to agents. In J. Horta & Y. Shoham (Eds.), *Proceedings of the 1993 AAAI Spring Symposium on Reasoning about Mental States: Formal Theories and Applications* (pp. 86-89). Menlo Park, CA: AAAI Press.

Barr, A. (1977). *Meta-knowledge and memory* (Tech. Rep. No. HPP-77-37). Stanford University, Department of Computer Science, Stanford, CA.

Barr, A. (1979). Meta-knowledge and cognition. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence* (pp. 31-33). Los Altos, CA: William Kaufmann.

Barsalou, L. W. (1983). Ad hoc categories. *Memory & Cognition*, 11, 211-227.

Barsalou, L. W. (1995). Storage side effects: Studying processing to understand learning. In A. Ram & D. B. Leake (Eds.), *Goal-driven learning* (pp. 407-419). Cambridge, MA: MIT Press/Bradford Books.

Barsalou, L. W., Hale, C. H., & Cox, M. T. (1989). *MECH: A computer interface for teaching and investigating mental models and troubleshooting* (Tech. Rep. No. GIT-ICS-89/17). Atlanta: Georgia Institute of Technology, College of Computing.

Batali, J. (1983). *Computational Introspection* (Tech. Rep. No. 701). Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Boston, MA.

Bhatta, S. (1995). *Model-based analogy in innovative device design*. Doctoral dissertation. Georgia Institute of Technology, College of Computing, Atlanta.

Bhatta, S., & Goel, A. (1992). Use of mental models for constraining index learning in experience-based design. In *Proceedings of the AAAI-92 Workshop on Constraining Learning with Prior Knowledge*.

Birnbaum, L. (1986). *Integrated processing in planning and understanding* (Tech. Rep. No. 489). Doctoral dissertation, Yale University, Department of Computer Science, New Haven, CT.

Birnbaum, L. (1989). Panel discussion on "indexing vocabulary." In *Case-Based Reasoning: Proceedings of a Workshop on Case-Based Reasoning* (p. 46). San Mateo, CA: Morgan Kaufmann.

Birnbaum, L. (1991). Rigor mortis: A response to Nilsson's 'Logic and artificial intelligence'. *Artificial Intelligence*, 47, 57-77.

Birnbaum, L., & Collins, G. (1984). Opportunistic planning and freudian slips. In *Proceedings of the Sixth Annual Conference of the Cognitive Science Society* (pp. 124-127). Hillsdale, NJ: Lawrence Erlbaum Associates.

Birnbaum, L., Collins, G., Freed, M., & Krulwich, B. (1990). Model-based diagnosis of planning failures. In *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 318-323). Menlo Park, CA: AAAI Press.

Booker, L. B., Goldberg, D. E. & Holland, J. H. (1989). Classifier systems and genetic algorithms. *Artificial Intelligence*, 40, 235-282.

Boring, E. G. (1953). A history of introspection. *Psychological Bulletin*, 50(3), 169-189.

Brazdil, P. B., & Konolige, K. (Eds.). (1990). *Machine learning, meta-reasoning and logics*. Norwell, MA: Kluwer Academic.

Brigham, M. C., & Pressley, M. (1988). Cognitive monitoring and strategy choice in younger and older adults. *Psychology and Aging*, 3(3), 249-257.

Brodley, C. (1993). Addressing the selective superiority problem: Automatic algorithm / model class selection. *Machine Learning: Proceedings of the Tenth International Conference* (pp. 17-24). San Mateo, CA: Morgan Kaufmann.

Brown, A. (1987). Metacognition, executive control, self-regulation, and other more mysterious mechanisms. In F. E. Weinert & R. H. Kluwe (Eds.), *Metacognition, motivation, and understanding* (pp. 65-116). Hillsdale, NJ: Lawrence Erlbaum Associates.

Brown, L. (1971). Untitled. In B. Ram Das, *Remember: Be here now*. San Cristobal, NM: Lama Foundation.

Buchanan, B. G., & Smith, R. G. (1989). Chapter XVIII: Fundamentals of expert systems. In A. Barr, P. R. Cohen, & E. A. Feigenbaum (Eds.), *The Handbook of artificial intelligence* (Vol. IV, pp. 149-192). Reading, MA: Addison-Wesley Publishing.

Carbonell, J. G. (1981). *Subjective understanding: Computer models of belief systems*. Ann Arbor, MI: UMI Research Press.

Carbonell, J. G. (1983). Learning by analogy: Formulating and generalizing plans from past experience. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning I: An artificial intelligence approach* (pp. 137-161). Los Altos, CA: Morgan Kaufmann.

Carbonell, J. G. (1986). Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In R. Michalski, J. Carbonell & T. Mitchell (Eds.), *Machine learning: An artificial intelligence approach, Vol. 2* (pp. 371-392). San Mateo, CA: Morgan Kaufmann Publishers.

Carbonell, J. G., & Gil, Y. (1990). Learning by experimentation: The operator refinement method. In Y. Kodratoff & R. S. Michalski (Eds.), *Machine learning III: An artificial intelligence approach* (pp. 191-213). San Mateo, CA: Morgan Kaufmann.

Carbonell, J. G., Knoblock, C. A., & Minton, S. (1991). PRODIGY: An integrated architecture for planning and learning. In K. VanLehn (Ed.), *Architectures for intelligence: The 22nd Carnegie Mellon symposium on cognition* (pp. 241-278). Hillsdale, NJ: Lawrence Erlbaum Associates.

Carey, S. (1986). Constraints on semantic development. In W. Demopoulos & A. Marras (Eds.), *Language learning and concept acquisition: Foundational issues*. (pp. 154-172). Norwood, NJ: Ablex Publishing.

Ceci, S. J. (Ed.) (1987). *Handbook of cognitive, social, and neuropsychological aspects of learning disabilities* (Vol. 2). Hillsdale, NJ: Lawrence Erlbaum Associates.

Celan, P. (1970). Untitled. In *Lichtzwang [Force of Light]* (p. 41). Berlin: Suhrkamp Verlag.

Celan, P. (1986). Untitled. In K. Washburn & M. Guillemin (Trans.), *Last poems* (p.39). San Francisco: North Point Press.



- Chalmers, D. (in press). *The conscious mind*. Oxford: Oxford University Press.
- Chandrasekaran, B. (1989). Task-structures, knowledge acquisition and learning, *Machine Learning*, 4, 339-345.
- Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., & Freeman, D. (1988). AUTO-CLASS: A Bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning* (pp. 54-64). San mateo, CA: Morgan Kaufmann.
- Cheng, J. (1995). *Management of speedup mechanisms in learning architectures* (Tech. Rep. No. CMU-CS-95-112). Doctoral dissertation, Pittsburgh: Carnegie Mellon University, School of Computer Science.
- Chi, M. T. H. (1987). Representing knowledge and metaknowledge: Implications for interpreting metamemory research. In F. E. Weinert & R. H. Kluwe (Eds.), *Metacognition, motivation, and understanding* (pp. 239-266). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Chi, M. T. H. (1995). *Revising the mental model as one learns*. Plenary address to the Seventeenth Annual Conference of the Cognitive Science Society. Pittsburgh (July 23).
- Chi, M. T. H., Bassok, M., Lewis, M., Reimann, P., & Glasser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, 13, 145-182.
- Chi, M. T. H., & VanLehn, K. A. (1991). The content of physics self-explanations. *The Journal of the Learning Sciences*, 1 (1) 69-105.
- Chinn, C. A., & Brewer, W. F. (1993). Factors that influence how people respond to anomalous data. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society* (pp. 318-323). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Clancey, W. J. (1987). *The knowledge engineer as student: Metacognitive bases for asking good questions* (Tech. Rep. No. STAN-CS-87-1183) Stanford University, Department of Computer Science, Stanford, CA.
- Clancey, W. J. (1991). The frame of reference problem in the design of intelligent machines. In K. VanLehn (Ed.), *Architectures for intelligence: The 22nd Carnegie Mellon symposium on cognition* (pp. 357-423). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Clancey, W. J. (1992). Model construction operators. *Artificial Intelligence*, 53, 1-115.
- Clancey, W. J. (1994). The biology of consciousness: Comparative review of Israel Rosenfield, *The strange, familiar, and forgotten: An anatomy of consciousness* and Gerald M.

Edelman, *Bright air, brilliant fire: On the matter of the mind*. In W. J. Clancey, S. W. Smoliar, & M. J. Stefik (Eds.), *Contemplating minds: A forum for artificial intelligence* (pp. 471- 514). Cambridge, MA: MIT Press.

Clancey, W. J., & Bock, C. (1985). *Representing control knowledge as abstract task and metarules*. (Tech. Rep. No. STAN-CS-87-1168) Stanford University, Department of Computer Science, Stanford, CA.

Cohen, S. M. (1990). *A model of troubleshooting in electronics assembly manufacturing* (Tech. Rep. No. CHMSR 90-3). Masters dissertation, Georgia Institute of Technology, Center for Human-Machine Systems Research, Atlanta.

Cohen, S. M., Mitchell, C. M., & Govindaraj, T. (1992). Analysis and aiding the human operator in electronics assembly. In M. Helander & M. Nagamachi (Eds.), *Design for manufacturability: A systems approach to concurrent engineering and ergonomics*. London: Taylor and Francis.

Coleridge, S. T. (1926). *Biographia Literaria*. New York: MacMillan. (Original work published 1817)

Collins, G. (1987). *Plan creation: Using strategies as blueprints* (Tech. Rep. No. 599). Doctoral dissertation, Yale University, Department of Computer Science, New Haven, CT.

Collins, G., Birnbaum, L., Krulwich, B., & Freed, M. (1993). The role of self-models in learning to plan. In A. Meyrowitz (Ed.), *Machine Learning: Induction, analogy and discovery*. Boston: Kluwer Academic Publishers.

Converse, T., & Hammond, K. J. (1992). Preconditions and appropriateness conditions. In *Proceedings of Fourteenth Annual Conference of the Cognitive Science Society* (pp. 13-17). Hillsdale, NJ: Lawrence Erlbaum Associates.

Cox, M. T. (1991). *Reasoning about reasoning via META-XP's*. Unpublished.

Cox, M. T. (1992). *Toward an epistemological treatment of the blame assignment problem*. Unpublished.

Cox, M. T. (1993). *Introspective multistrategy learning*. (Tech. Rep. No. GIT-COGSCI-94/02). Atlanta: Georgia Institute of Technology, College of Computing.

Cox, M. T. (1994a). Case-based introspection [Research summary]. In *Proceedings of the 12th National Conference on Artificial Intelligence* (p. 1435). Menlo Park, CA: AAAI Press.

Cox, M. T. (1994b). Machines that forget: Learning from retrieval failure of mis-indexed explanations. In A. Ram & K. Eiselt (Eds.), *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society* (pp. 225-230). Hillsdale, NJ: Lawrence Erlbaum Associates.

Cox, M. T. (1994c). *Metacognition, problem solving and aging*. (Tech. Rep. No. GIT-COGSCI-94/15). Atlanta: Georgia Institute of Technology, College of Computing.

Cox, M. T. (1995). Representing mental events (or the lack thereof). In M. T. Cox & M. Freed (Eds.), *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms* (pp. 22-30). Menlo Park, CA: AAAI Press. (Available as Tech. Rep. No. SS-95-08)

Cox, M. T. & Freed, M. (1994). Using knowledge from cognitive behavior to learn from failure. In J. W. Brahan & G. E. Lasker (Eds.), *Proceedings of the Seventh International Conference on Systems Research, Informatics and Cybernetics: Vol. 2. Advances in Artificial Intelligence - Theory and Application II* (pp. 142-147). Windsor, Ontario, Canada: The International Institute for Advanced Studies in Systems Research and Cybernetics.

Cox, M. T. & Freed, M. (Eds.). (1995). *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms* (Tech. Rep. No. SS-95-08). Menlo Park, CA: AAAI Press.

Cox, M. T., & Kell, A. (1993). *Reflective learning in the domain of programming*. Unpublished.

Cox, M. T., & Ram, A. (1991). Using introspective reasoning to select learning strategies. In R. S. Michalski & G. Tecuci (Eds.), *Proceedings of the First International Workshop on Multistrategy Learning* (pp. 217-230). Washington, DC: George Mason University, Artificial Intelligence Center.

Cox, M. T., & Ram, A. (1992a). An explicit representation of forgetting. In J. W. Brahan & G. E. Lasker (Eds.), *Proceedings of the Sixth International Conference on Systems Research, Informatics and Cybernetics: Vol. 2. Advances in Artificial Intelligence - Theory and Application* (pp. 115-120). Windsor, Ontario, Canada: The International Institute for Advanced Studies in Systems Research and Cybernetics.

Cox, M. T., & Ram, A. (1992b). Multistrategy learning with introspective meta-explanations. In D. Sleeman & P. Edwards (Eds.), *Machine Learning: Proceedings of the Ninth International Conference* (pp. 123-128). San Mateo, CA: Morgan Kaufmann.

Cox, M. T., & Ram, A. (1994a). Choosing learning strategies to achieve learning goals. In M. desJardins & A. Ram (Eds.), *Proceedings of the 1994 AAAI Spring Symposium on Goal-*

*Driven Learning* (pp. 12-21). Menlo Park, CA: AAAI Press.

Cox, M. T., & Ram, A. (1994b). Failure-driven learning as input bias. In A. Ram & K. Eiselt (Eds.), *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society* (pp. 231-236). Hillsdale, NJ: Lawrence Erlbaum Associates.

Cox, M. T., & Ram, A. (1995). Interacting learning-goals: Treating learning as a planning task. In J.-P. Haton, M. Keane & M. Manago (Eds.), *Advances in case-based reasoning* (pp. 60-74). Berlin: Springer-Verlag.

Crockett, L. J. (1994). *The Turing Test and the frame problem: AI's mistaken understanding of intelligence*. Norwood, NJ: Ablex Publishing.

Cullingford, R. (1978). *Script application: Computer understanding of newspaper stories* (Tech. Rep. No. 116). Doctoral dissertation, Yale University, Department of Computer Science, New Haven, CT.

Cullingford, R. (1981). Micro SAM. In R. C. Schank & C. Riesbeck (Eds.), *Inside computer understanding: Five programs plus miniatures* (pp. 120-135). Hillsdale, NJ: Lawrence Erlbaum Associates.

Danyluk, A. P. (1994). GEMINI: An integration of analytical and empirical learning. In R. S. Michalski & G. Tecuci (Eds.), *Machine learning IV: A multistrategy approach* (pp. 189-215). San Francisco: Morgan Kaufmann.

Davidson, J. E., Deuser, R., & Sternberg, R. J. (1994). The role of metacognition in problem solving. In J. Metcalfe & A. P. Shimamura (Eds.), *Metacognition: Knowing about knowing* (pp. 207-226). Cambridge, MA: MIT Press/Bradford Books.

Davis, R. (1979). Interactive transfer of expertise: Acquisition of new inference rules. *Artificial Intelligence*, 12, 121-157).

Davis, R. (1980). Meta-rules: Reasoning about control. *Artificial Intelligence*, 15, 179-222.

Davis, R., & Buchanan, B. G. (1977). Meta-level knowledge: Overview and applications. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (Vol. 2, pp. 920-927). Pittsburgh: Carnegie-Mellon University, Department of Computer Science.

DeJong, G., & Mooney, R. (1986). Explanation-based learning: An alternative view. *Machine Learning*, 1(2), 145-176.

- de Kleer, J., & Brown, J. S. (1988). Assumptions and ambiguities in mechanistic mental models. In A. Collins & E. E. Smith (Eds.), *Readings in cognitive science: A perspective from psychology and artificial intelligence* (pp. 270-287). San Mateo, CA: Morgan Kaufmann. (Original work published 1983)
- deKleer, J., Doyle, J., Steele, G. L., & Sussman, G. J. (1977). Explicit control of reasoning. *SIGPLAN Notices*, 12.
- Delclos, V. R., & Harrington, C. (1991). Effects of strategy monitoring and proactive instruction on children's problem-solving performance. *Journal of Educational Psychology*, 83(1), 35-42.
- Dennett, D. C. (1978). *Brainstorms: Philosophical essays on mind and psychology*. Cambridge, MA: MIT Press/Bradford Books.
- Dennett, D. C. (1992). *Consciousness explained*. Boston: Little, Brown & Company.
- Derry, S. J. (1989). Strategy and expertise in solving word problems. In C. B. McCormick, G. E. Miller, & M. Pressley (Eds.), *Cognitive strategy research: From basic research to educational applications* (pp. 269-302). Berlin: Springer-Verlag.
- Descartes, R. (1955). Discourse on the method of rightly conducting the reason and seeking for the truth in the sciences. In E. S. Haldane & G. R. T. Ross (Trans.), *The philosophical works of Descartes* (Vol. 1, pp. 79-130). Dover Publications. (Original work published 1637)
- Descartes, R. (1955). The passions of the soul. In E. S. Haldane & G. R. T. Ross (Trans.), *The philosophical works of Descartes* (Vol. 1, pp. 329-427). Dover Publications. (Original work published 1649)
- desJardins, M. (1992). Goal-directed learning: A decision-theoretic model for deciding what to learn next. In *Proceedings of the ML-92 Workshop on Machine Discovery* (pp. 147-151).
- Domeshek, E. A. (1992). *Do the right thing: A component theory for indexing stories as social advice* (Tech. Rep. No. 26). Doctoral dissertation, Northwestern University, Institute for the Learning Sciences, Evanston, IL.
- Dörner, D. (1979). Self-reflection and problem-solving. In F. Klix (Ed.), *Human and artificial intelligence* (pp. 101-107). Amsterdam: North Holland.
- Doyle, J. (1979). A truth maintenance system, *Artificial Intelligence*, 12, 231-272.

Doyle, J. (1980). *A model for deliberation, action, and introspection* (Tech. Rep. No. TR-581). Doctoral dissertation, Massachusetts Institute of Technology, Department of Computer Science, Cambridge.

Edelman, G. M. (1992). *Bright air, brilliant fire: On the matter of the mind*. New York: Basic Books.

Etzioni, O. (1991). Embedding decision-analytic control in a learning architecture. *Artificial Intelligence*, 49, 129-159.

Etzioni, O., Hanks, S., Weld, D., Draper, D., Lesh, N., & Williamson, M. (1992). An approach to planning with incomplete information. In *Proceedings of the Third International Conference on Principles of Knowledge representation and reasoning* (pp. 115-125). San Mateo, CA: Morgan Kaufmann.

Fikes, R. E., & Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 3, 251-288.

Fischhoff, B., Slovic, P., & Lichtenstein, S. (1977). Knowing with certainty: The appropriateness of extreme confidence. *Journal of Experimental Psychology: Human Perception and Performance*, 3(4), 552-564.

Flavell, J. H. (1971). First discussant's comments: What is memory development the development of? *Human Development*, 14, 272-278.

Flavell, J. H., & Wellman, H. M. (1977). Metamemory. In R. V. Kail, Jr., & J. W. Hagen (Eds.), *Perspectives on the Development of Memory and Cognition* (pp. 3-33). Hillsdale, NJ: Lawrence Erlbaum Associates.

Forrest-Pressley, D. L., MacKinnon, G. E., & Waller, T. G. (Eds.) (1985). *Metacognition, cognition and human performance* (Vol. 2, Instructional practices). New York: Academic Press.

Fox, S. (1995). *Introspective learning for case-based planning*. Unpublished doctoral dissertation, Indiana University, Department of Computer Science, Bloomington, IN.

Fox, S., & Leake, D. (1994). Using introspective reasoning to guide index refinement in case-based reasoning. In A. Ram & K. Eiselt (Eds.), *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society* (pp. 324-329). Hillsdale, NJ: Lawrence Erlbaum Associates.

Fox, S., & Leake, D. (1995a). Modeling case-based planning for repairing reasoning failures. In M. T. Cox & M. Freed (Eds.), *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms* (pp. 31-38). Menlo Park, CA: AAAI Press.

(Available as Tech. Rep. No. SS-95-08)

Fox, S., & Leake, D. (1995b). Using introspective reasoning to refine indexing. In C. S. Mellish (Ed.), *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* (pp. 391-397). San Mateo, CA: Morgan Kaufmann.

Freed, M., & Collins, G. (1994). Learning to cope with task interactions. In A. Ram & M. desJardins (Eds.), *Proceedings of the 1994 AAAI Spring Symposium on Goal-Driven Learning* (pp. 28-35). Menlo Park, CA: AAAI Press.

Freed, M., Krulwich, B., Birnbaum, L., & Collins, G. (1992). Reasoning about performance intentions. In *Proceedings of Fourteenth Annual Conference of the Cognitive Science Society* (pp. 7-12). Hillsdale, NJ: Lawrence Erlbaum Associates.

Gardner, H. (1987). *The mind's new science: A history of the cognitive revolution*. New York: Basic Books.

Garner, R. (1987). *Metacognition and reading comprehension*. Norwood, NJ: Ablex Publishing Corporation.

Gavelek, J. R., & Raphael, T. E. (1985). Metacognition, instruction, and the role of questioning activities. In D. L. Forrest-Pressley, G. E. MacKinnon, and T. G. Waller (Eds.), *Metacognition, Cognition and Human Performance*. Vol. 2 (Instructional Practices), Academic Press, Inc., New York, pp. 103-136.

Genesereth, M. R. (1983). An overview of meta-level architecture. In *Proceedings of the Third National Conference on Artificial Intelligence* (pp. 119-123). Los Altos, CA: William Kaufmann.

Ghosh, S., Hendler, J., Kambhampati, S., & Kettler, B. (1992). *UM Nonlin* [a Common Lisp implementation of A. Tate's Nonlin planner]. Available FTP: Hostname: cs.umd.edu Directory: /pub/nonlin Files: nonlin-files.tar.Z

Glenberg, A. M., Wilkinson, A. C., & Epstein, W. (1992). The illusion of knowing: Failure in the self-assessment of comprehension. In T. O. Nelson (Ed.), *Metacognition: Core readings* (pp. 185-195). Boston: Allyn and Bacon. (Original work published 1982)

Goel, A. K., Ali, K. S., Donnellan, M. W., Garza, A. G., & Callantine, T. J. (1994). Multi-strategy adaptive path planning. *IEEE Expert*, 9(6), 57-65.

Gombert, J. E. (1992). *Metalinguistic development*. Chicago: University of Chicago Press.

Green, C. C. (1969). The application of theorem proving to question answering. In *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 219-237). Los Altos, CA: Kaufmann.

Greeno, J. G. (1977). Process of understanding in problem solving. In N. J. Castellan, D. B. Pisoni, & G. R. Potts (Eds.), *Cognitive theory* (Vol. 2). Hillsdale, NJ: Lawrence Erlbaum Associates.

Greeno, J. G., & Riley, M. S. (1987). Processes and development of understanding In F. E. Weinert & R. H. Kluwe (Eds.), *Metacognition, motivation, and understanding* (pp. 289-313). Hillsdale, NJ: Lawrence Erlbaum Associates.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), 199-220.

Hale, C. R., & Barsalou, L. W. (1995). Explanation content and construction during system learning and troubleshooting. *The Journal of the Learning Sciences*, 4(4), 385-436.

Hammond, K. J. (1986). Learning to anticipate and avoid planning problems through the explanation of failures. In *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 556-560). Los Altos, CA: Morgan Kaufmann.

Hammond, K. J. (1988). Opportunistic memory: Storing and recalling suspended goals. In J. L. Kolodner (Ed.), *Proceedings of a Workshop on Case-Based Reasoning* (pp. 154-168). San Mateo, CA: Morgan Kaufmann.

Hammond, K. J. (1989). *Case-based planning: Viewing planning as a memory task. Vol. 1. of Perspectives in artificial intelligence*. San Diego, CA: Academic Press.

Hammond, K. J. (1990). Learning and enforcement: Stabilizing environments to facilitate activity. In B. W. Porter & R. J. Mooney (Eds.), *Machine Learning: Proceedings of the Seventh International Conference* (pp. 204-210). San Mateo, CA: Morgan Kaufmann.

Hammond, K., Converse, T., Marks, M., & Seifert, C. (1993). Opportunism and learning. In J. L. Kolodner (Ed.), *Case-based learning* (pp. 85-115). Boston: Kluwer Academic.

Handey, J. (1992). *Deep Thoughts*. New York: Berkley Publishing Group.

Hayes, P. J. (1992). The naïve physics manifesto. In M. A. Boden (Ed.), *The philosophy of artificial intelligence* (pp. 171-205). Oxford: Oxford University Press. (Original work published 1979)



- Hayes, P. J. (1981). The logic of frames. In B. L. Webber & N. J. Nilsson (Eds.), *Readings in artificial intelligence* (pp. 451-458). Los Altos, CA: Morgan Kaufmann. (Original work published 1979)
- Hayes, P. J., Ford, K. M., & Agnew, N. (1994). On babies and bathwater: A cautionary tale. *AI Magazine*, 15(4), 15-26.
- Hayes-Roth, B., & Hayes-Roth, F. (1979). A cognitive model of planning, *Cognitive Science*, 2, 275-310.
- Hayes-Roth, F. (1983). Using proofs and refutations to learn from experience. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine Learning I: An artificial intelligence approach* (pp. 221-240). Los Altos, CA: Morgan Kaufmann.
- Hayes-Roth, F., Waterman, D. A., & Lenat, D. B. (Eds.). (1983). *Building expert systems*. London: Addison-Wesley Publishing.
- Heinlein, R. A. (1966). *The moon is a harsh mistress*. New York: Putnam.
- Hinrichs, T. R. (1992). *Problem solving in open worlds*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Hofstadter, D. R. (1989). *Gödel, Escher, Bach: An eternal golden braid*. New York: Vintage Books. (Original work published in 1979)
- Horty, J., & Shoham, Y. (Eds.). (1993). *Proceedings of the 1993 AAAI Spring Symposium on Reasoning about Mental States: Formal Theories and Applications*. Menlo Park, CA: AAAI Press.
- Hume, G., Michael, J., Rovick, A., & Evens, M. (1996). Hinting as a tactic in one-on-one tutoring. *The Journal of the Learning Sciences*, 5(1), 23-47.
- Hunter, L. E. (1989). *Knowledge acquisition planning: Gaining experience through experience* (Tech. Rep. No. 678). Doctoral dissertation, Yale University, Department of Computer Science, New Haven, CT.
- Hunter, L. E. (1990a). Knowledge acquisition planning for inference from large datasets. In *Proceedings of the Twenty Third Annual Hawaii International Conference on System Sciences* (pp. 35-44). Washington, DC: IEEE Press.
- Hunter, L. E. (1990b). Planning to learn. In *Proceedings of Twelfth Annual Conference of the Cognitive Science Society* (pp. 261-276). Hillsdale, NJ: Lawrence Erlbaum Associates.

Hultsch, D. F., Hertzog, C. K., Dixon, R. A. & Davidson, H. (1988). Memory self-knowledge and self-efficacy in the aged. In M. L. Howe & C. J. Brainerd (Eds.), *Cognitive Development in Adulthood* (pp. 65-92). New York: Springer-Verlag.

Jameson, A., Nelson, T. O., Leonesio, R. J., & Narens, L. (1993). The feeling of another person's knowing. *Journal of Memory and Language*, 32, 320-335.

Johnson, H. M., & Seifert, C. M. (in press). Sources of the continued influence effect: When misinformation in memory affects later inferences. *Journal of Experimental Psychology: Learning, Memory, & Cognition*.

Johnson-Laird, P. N. (1988). *The computer and the mind: An introduction to cognitive science*. Cambridge, MA: Harvard University Press.

Johnson-Laird, P. N. (1983). *Mental models: Toward a cognitive science of language, inference, and consciousness*. Cambridge: Cambridge University Press.

Jones, E. K. (1992). *The flexible use of abstract knowledge in planning*. Doctoral dissertation, Yale University, Department of Computer Science, New Haven, CT. (Available as Tech. Rep. No. 28, Northwestern University, Institute for the Learning Sciences, Evanston, IL)

Jones, R. M., & VanLehn, K. (1991). Strategy shifts without impasses: A computational model of the sum-to-min transition. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society* (pp. 358-363). Hillsdale, NJ: Lawrence Erlbaum Associates.

Kass, A. (1986). Modifying explanations to understand stories. In *Proceedings of Eighth Annual Conference of the Cognitive Science Society* (pp. 691-696). Hillsdale, NJ: Lawrence Erlbaum Associates.

Kass, A. (1990). *Developing creative hypotheses by adapting explanations*. Doctoral dissertation, Northwestern University, The Institute for the Learning Sciences, Evanston, IL.

Kass, A., Leake, D., & Owens, C. (1986). SWALE: A program that explains, In R. C. Schank, *Explanation patterns: Understanding mechanically and creatively*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Kausler, D. H. (1991). *Experimental psychology, cognition, and human aging* (2nd ed.). New York: Springer-Verlag.

Keil, F. C., (1981) Constraints on knowledge and cognitive development. *Psychology Review*, 88(3), 197- 227.

- Keil, F. C. (1989) *Concepts, kinds, and conceptual development*. Cambridge MA: MIT Press
- Keller, R. M. (1986). *Deciding what to learn* (Tech. Rep. No. ML-TR-6). Rutgers University, Department of Computer Science.
- Kennedy, A. C. (1995). Using a domain-independent introspection mechanism to improve memory search. In M. T. Cox & M. Freed (Eds.), *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms* (pp. 72-78). Menlo Park, CA: AAAI Press. (Available as Tech. Rep. No. SS-95-08)
- Kerner, Y. (1995). Case-based evaluation in computer chess. In J.-P. Haton, M. Keane, & M. Manago (Eds.), *Advances in case-based reasoning* (pp. 240-254). Berlin: Springer-Verlag.
- Klahr, D. & Dunbar, K. (1988). Dual space search during scientific reasoning. *Cognitive Science*, 12, 1-48.
- Kluwe, R. H. (1987). Executive decisions and regulation of problem solving behavior. In F. E. Weinert & R. H. Kluwe (Eds.), *Metacognition, motivation, and understanding* (pp. 31-64). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Kolodner, J. L. (1984). *Retrieval and organizational strategies in conceptual memory: A computer model*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Kolodner, J. L. (1987). Capitalizing on failure through case-based inference. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society* (pp. 715-726) Hillsdale, NJ: Lawrence Erlbaum Associates.
- Kolodner, J. L. (1993). *Case-based reasoning*. San Mateo, CA: Morgan Kaufmann Publishers.
- Kolodner, J. L., & Simpson, R. L. (1989). The MEDIATOR: Analysis of an early case-based problem solver. *Cognitive Science*, 13, 507-549.
- Konolige, K. (1985). A computational theory of belief introspection. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 502-508). Los Altos, CA: Morgan Kaufmann.
- Konolige, K. (1988). Reasoning by introspection. In P. Maes & D. Nardi (Eds.), *Meta-level architectures and reflection* (pp. 61-74). Amsterdam: North-Holland.

Koton, P. (1989). *Using experience in learning and problem solving*. Doctoral dissertation, Massachusetts Institute of Technology, Cambridge.

Krinsky, R., & Nelson, T. O. (1985). The feeling of knowing for different types of retrieval failure. *Acta Psychologica*, 58, 141-158.

Krulwich, B. (1991). Determining what to learn in a multi-component planning system. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*. Chicago, IL, (August 7-10), pp. 102-107.

Krulwich, B. (1993). *Flexible learning in a multicomponent planning system*. (Tech. Rep. No. 46). Doctoral dissertation, Northwestern University, The Institute for the Learning Sciences, Evanston, IL.

Krulwich, B. (1995). Cognitive behavior, basic levels, and intelligent agents. In M. T. Cox & M. Freed (Eds.), *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms* (pp. 79-83). Menlo Park, CA: AAAI Press. (Available as Tech. Rep. No. SS-95-08)

Kuokka, D. R. (1990). *The deliberative integration of planning, execution, and learning* (Tech. Rep. No. CMU-CS-90-135). Doctoral dissertation, Carnegie Mellon University, Pittsburgh.

Lachman, J. L., Lachman, R., & Thronesbery, C. (1979). Metamemory through the adult life span. *Developmental Psychology*, 15(5), 543-551.

Laird, J. E., Rosenbloom, P. S., & Newell, A. (1986). Chunking in Soar: The anatomy of a general learning mechanism, *Machine Learning*, 1, 11-46.

Leake, D. (1992). *Evaluating explanations: A content theory*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Leake, D. (1995). Representing self-knowledge for introspection about memory search. In M. T. Cox & M. Freed (Eds.), *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms* (pp. 84-88). Menlo Park, CA: AAAI Press. (Available as Tech. Rep. No. SS-95-08)

Leake, D., & Ram, A. (1993). Goal-driven learning: Fundamental issues and symposium report. *AI Magazine*, 14(4), 67-72.

Lebowitz, M. (1987). Experiments with incremental concept formation: UMIMEM. *Machine Learning*, 2, 103-138.

- Lehnert, W., Dyer, M. G., Johnson, P., Yang, C., & Harley, S. (1983). BORIS - An experiment in in-depth understanding of narratives. *Artificial Intelligence*, 20(1), 15-62.
- Lehnert, W., Cardie, C., Fisher, D., McCarthy, J., Riloff, E., Soderland, S. (1995). Evaluating an information extraction system. *Journal of Integrated Computer-Aided Engineering*, 1(6), 453-472.
- Lehnert, W., & Sundheim, B. (1991, Fall). A performance evaluation of text analysis technologies. *AI Magazine*, pp. 81-94.
- Lenat, D. B., Davis, R., Doyle, J., Genesereth, M., Goldstein, I., & Schrobe, H. (1983). Reasoning about reasoning. In F. Hayes-Roth, D. A. Waterman, & D. B. Lenat (Eds.), *Building expert systems* (pp. 219-239). London: Addison-Wesley Publishing.
- Lieberman, D. A. (1992) Behaviorism and the mind: A (limited) call for a return to introspection. In T. O. Nelson (Ed.), *Metacognition: Core readings* (pp. 9-24). Boston: Allyn and Bacon. (Original work published 1979)
- Loebner, H. G. (1994). In Response (to Shieber's lessons from a restricted Turing test). *Communications of the ACM*, 37(6), 79-82.
- Lovelace, E. A. (1990). Aging and metacognitions concerning memory function. In Eugene A. Lovelace (Ed.), *Aging and cognition: Mental process, self-awareness and interventions* (pp. 157-188). Amsterdam: North Holland.
- Lovelace, E. A., & Marsh, G. R. (1985). Prediction and evaluation of memory performance by young and old adults. *Journal of Gerontology*, 40, 192-197.
- Lyons, W. (1986). *The disappearance of introspection*. Cambridge, MA: Bradford Books/MIT Press.
- Maes, P. (1987a). *Computational reflection* (Tech. Rep. No. 87-2). Doctoral dissertation, Vrije Universiteit Brussels, Artificial Intelligence Laboratory, Brussels, Belgium.
- Maes, P. (1987b). Introspection in knowledge representation. In Du Boulay, B., Hogg, D., & Steels, L. (Eds.), *Advances in Artificial Intelligence - II* (pp. 249-262). Amsterdam: North-Holland.
- Maes, P. (1988). Issues in computational reflection. In P. Maes & D. Nardi (Eds.), *Meta-level architectures and reflection* (pp. 21-35). Amsterdam: North Holland.
- Maes, P., & Nardi, D. (Eds.). (1988). *Meta-level architectures and reflection*. Amsterdam: North-Holland.

Markovitch, S., & Scott, P. D. (1988). The role of forgetting in learning. In J. Laird (Ed), *Proceedings of the Fifth International Conference on Machine Learning* (pp. 459-465). San Mateo, CA: Morgan Kaufmann.

Markovitch, S., & Scott, P. D. (1993). Information filtering: Selection mechanisms in learning systems. *Machine Learning*, 10 (2), 113-151.

Martin, C. E. (1989). Micro DMAP. In C. K. Riesbeck & R. C. Schank (Eds.), *Inside case-based reasoning* (pp. 353-372). Hillsdale, NJ: Lawrence Erlbaum Associates.

Martin, C. E. (1990). *Direct memory access parsing* (Tech. Rep. No. 93-07). Doctoral dissertation, University of Chicago, Department of Computer Science, Chicago.

Martin, J. (1992, December). *Personal communication*.

McCarthy, J. (1959). Programs with common sense. In *Symposium Proceedings on Mechanisation of Thought Processes* (Vol. 1, pp. 77-84). London: Her Majesty's Stationary Office.

McCarthy, J. (1968). Programs with common sense. In M. L. Minsky (Ed.), *Semantic information processing* (pp. 403-418). Cambridge, MA: MIT Press.

McCarthy, J. (1979). Ascribing mental qualities to machines. In M. Ringle (Ed.), *Philosophical perspectives in artificial intelligence* (pp. 161-195). Atlantic Highlands, NJ: Humanities Press.

McCarthy, J. (1993). Notes on formalizing context. In R. Bajcsy (Ed.), *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence* (Vol. 1, pp.555-560). San Mateo, CA: Morgan Kaufmann.

McCarthy, J. (1995). Making robots conscious of their mental states. In M. T. Cox & M. Freed (Eds.), *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms* (pp. 89-96). Menlo Park, CA: AAAI Press. (Available as Tech. Rep. No. SS-95-08)

McCarthy, J., & Hayes, P. (1969). Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4, 463-502.

McDermott, D. (1989). *A general framework for reason maintenance* (Tech. Rep. No. 691). Yale University, Department of Computer Science, New Haven, CT.

McDermott, D. (1992). A critique of pure reason. In M. A. Boden (Ed.), *The philosophy of artificial intelligence* (pp. 206-230). Oxford: Oxford University Press. (Original work pub-

lished 1987)

McDermott, J. (1988). Preliminary steps toward a taxonomy of problem-solving methods. In S. Marcus (Ed.), *Automating knowledge acquisition for expert systems* (pp. 225- 256). Boston: Kluwer Academic Publishers.

McNamara, T. P., Miller, D. L., & Bransford, J. D. (1991). Mental models and reading comprehension. In R. Barr, M. L. Kamil, P. Mosenthal, & P. D. Pearson (Eds.), *Handbook of reading research* (Vol. 2, pp. 490- 511). New York: Longman.

McRoy, S. W. (1993). Belief as an effect of an act of introspection: Some preliminary remarks. In J. Horty & Y. Shoham (Eds.), *Proceedings of the 1993 AAAI Spring Symposium on Reasoning about Mental States: Formal Theories and Applications* (pp. 86-89). Menlo Park, CA: AAAI Press.

Meehan, J. (1981). Talespin. In R. C. Schank & C. Riesbeck (Eds.), *Inside computer understanding: Five programs plus miniatures* (pp. 197-258). Hillsdale, NJ: Lawrence Erlbaum Associates.

Metcalfe, J., & Shimamura, A. P. (Eds.). (1994). *Metacognition: Knowing about knowing*. Cambridge, MA: MIT Press/Bradford Books.

Michalski, R. S. (1991). Inferential learning theory as a basis for multistrategy task-adaptive learning. In R. S. Michalski & G. Tecuci (Eds.), *Proceedings of the First International Workshop on Multistrategy Learning* (pp. 3-18). Washington, DC: George Mason University, Artificial Intelligence Center.

Michalski, R. S. (Ed.). (1993). Multistrategy learning [Special issue]. *Machine Learning*, 11 (2/3).

Michalski, R. S. (1994). Inferential theory of learning: Developing foundations for multistrategy learning. In R. S. Michalski & G. Tecuci (Eds.), *Machine learning IV: A multistrategy approach* (pp. 3-61). San Francisco: Morgan Kaufmann.

Michalski, R. S., & Ram, A. (1995). Learning as goal-driven inference. In A. Ram & D. Leake (Eds.), *Goal-driven learning* (pp. 479-490). Cambridge, MA: MIT Press/Bradford Books.

Michalski, R. S., & Tecuci, G. (Eds.). (1991). *Proceedings of the First International Workshop on Multistrategy Learning*. Washington, DC: George Mason University, Artificial Intelligence Center.

Michalski, R. S. & Tecuci, G. (Eds.). (1994). *Machine learning IV: A multistrategy approach*. San Francisco: Morgan Kaufmann.

Miner, A. C., & Reder, L. M. (1994). A new look at feeling of knowing: Its metacognitive role in regulating question answering. In J. Metcalfe & A. P. Shimamura (Eds.), *Metacognition: Knowing about knowing* (pp. 47-70). Cambridge, MA: MIT Press/Bradford Books.

Minsky, M. L. (1963). Steps towards artificial intelligence. In E. A. Feigenbaum & J. Feldman (Eds.), *Computers and thought* (pp. 406-450). New York: McGraw Hill. (Original work published 1961)

Minsky, M. L. (1965). Matter, mind, and models. In *Proceedings of the International Federation of Information Processing Congress 1965* (Vol. 1, pp. 45-49).

Minsky, M. L. (1968a). Matter, mind, and models. In M. L. Minsky (Ed.), *Semantic information processing* (pp. 425-432). Cambridge, MA: MIT Press.

Minsky, M. L. (Ed.). (1968b). *Semantic information processing*. Cambridge, MA: MIT Press.

Minsky, M. L. (1975). A framework for representing knowledge. In P. H. Winston (Ed.), *The psychology of computer vision* (pp. 211-277). New York: McGraw Hill.

Minton, S. (1988). *Learning search control knowledge: A explanation-based approach*. Boston: Kluwer Academic.

Minton, S. (1990). Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, 42, 363-392.

Minton, S., Carbonell, J. G., Etzioni, O., Knoblock, C., & Kuokka, D. (1987). Acquiring effective search control rules: Explanation-based learning in the PRODIGY system. In P. Langley (Ed.), *Proceedings of the Fourth International Workshop on Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Mitchell, T. M. (1990). The need for biases in learning generalizations. In J.W. Shavlik & T.G. Dietterich (Eds.), *Readings in machine learning* (pp. 184-191). San Mateo, CA: Morgan Kaufmann. (Original work published 1980)

Mitchell, T. M., Allen, J., Chalasani, P., Cheng, J., Etzioni, O., Ringuette, M., & Schlimmer, J. C. (1991). Theo: A framework for self-improving systems. In K. VanLehn (Ed.), *Architectures of cognition: The 22nd Carnegie Mellon symposium on cognition* (pp. 323-355). Hillsdale, NJ: Lawrence Erlbaum Associates.



- Mitchell, T. M., Keller, R., & Kedar-Cabelli, S. (1986). Explanation-based generalization: A unifying view, *Machine Learning*, 1(1), 47-80.
- Mitchell, T. M., Utgoff, P. E., & Banerji, R. (1983). Learning by experimentation: Acquiring and refining problem-solving heuristics. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine Learning I: An artificial intelligence approach* (pp. 163-189). Los Altos, CA: Morgan Kaufmann.
- Mooney, R. (1993). Integrating theory and data in category learning. In G. Nakamura, D. Medin, & R. Taraban (Eds.), *The psychology of learning and motivation (Vol. 29): Categorization by humans and machines* (pp. 189-218). New York: Academic Press.
- Mooney, R., & Ourston, D. (1991). Improving shared rules in multiple category domain theories. In *Proceedings of the Eighth International Workshop on Machine Learning* (pp. 534-538). San Mateo, CA: Morgan Kaufmann.
- Mooney, R., & Ourston, D. (1994). A multistrategy approach to theory refinement. In R. S. Michalski & G. Tecuci (Eds.), *Machine learning IV: A multistrategy approach* (pp. 141-164). San Francisco: Morgan Kaufmann.
- Moore, A. W., & Lee, M. S. (1994) Efficient algorithms for minimizing cross validation error. In W. W. Cohen & H. Hirsh (Eds.), *Machine Learning: Proceedings of the Eleventh International Conference* (pp. 190-198). San Francisco: Morgan Kaufmann.
- Moore, R. C. (1977). Reasoning about knowledge and action. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence* (Vol. 1, pp. 223-227). Pittsburgh: Carnegie-Mellon University, Department of Computer Science.
- Moore, R. C. (1995). *Logic and representation*. Stanford, CA: CSLI Publications.
- Moorman, K., & Ram, A. (1994a). A model of creative understanding. In *Proceedings of the 12th National Conference on Artificial Intelligence* (pp. 74-79). Menlo Park, CA: AAAI Press.
- Moorman, K., & Ram, A. (1994b). Integrating creativity and reading: A functional approach. In A. Ram & K. Eiselt (Eds.), *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society* (pp. 646-651). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Morik, K. (1994). Balanced cooperative modeling. In R. S. Michalski & G. Tecuci (Eds.), *Machine learning IV: A multistrategy approach* (pp. 295-317). San Francisco: Morgan Kaufmann.

Narayanan, S., Ram, A., Cohen, S. M., Mitchell, C. M., & Govindaraj, T. (1992). Knowledge-based diagnostic problem solving and learning in the test area of electronics assembly manufacturing. In *Proceedings of the SPIE Conference on Applications of AI X: Knowledge-Based Systems*. Orlando, FL.

Neisser, U. (Ed.). (1993). *The perceived self: Ecological and interpersonal sources of self-knowledge*. New York: Cambridge University Press.

Neisser, U. (1995, October). *The future of cognitive psychology: An ecological analysis*. Cognitive Science Colloquium Address, College of Computing, Georgia Institute of Technology, Atlanta.

Nelson, T. O., & Dunlosky, J. (1991). When people's judgements of learning (JOLs) are extremely accurate at predicting subsequent recall: The "Delayed-JOL Effect." *Psychological Science*, 2(4), 267-270.

Nelson, T. O., & Narens, L. (1992). Metamemory: A theoretical framework and new findings. In T. O. Nelson (Ed.), *Metacognition: Core readings* (pp. 9-24). Boston: Allyn and Bacon. (Originally published in 1990.)

Nelson, T. O., & Narens, L. (1994). Why investigate metacognition? In J. Metcalfe & A. P. Shimamura (Eds.), *Metacognition: Knowing about knowing* (pp. 1-25). Cambridge, MA: MIT Press/Bradford Books.

Nersessian, N. J. (1992). How do scientists think? Capturing the dynamics of conceptual change in science. In R. Giere (Ed.), *Cognitive models of science (Minnesota Studies in the Philosophy of Science 15)*. Minneapolis: University of Minnesota Press.

Newell, A. (1982). The knowledge level. *Artificial Intelligence*, 18, 87-127.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Newell, A., & Simon, H. A. (1963). GPS, a program that simulates human thought. In E. A. Feigenbaum & J. Feldman (Eds.), *Computers and thought* (pp. 279-293). New York: McGraw Hill.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Ng, E., & Bereiter, C. (1991). Three levels of goal-orientation in learning. *The Journal of the Learning Sciences*, 1(3/4), 243-271.

- Nisbett, R. E. & Wilson, T. (1977). Telling more than we can know: Verbal reports on mental processes. *Psychological Review*, 84(3), 231-259.
- Norman, D. A. (1979). Analysis and design of intelligent systems. In F. Klix (Ed.), *Human and artificial intelligence* (pp. 37-43). Amsterdam: North Holland.
- Oehlmann, R., Edwards, P., & Sleeman, D. (1994). Changing the viewpoint: Re-indexing by introspective questioning. In A. Ram & K. Eiselt (Eds.), *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society* (pp. 675-680). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Oehlmann, R., Edwards, P., & Sleeman, D. (1995). Introspection planning: Representing metacognitive experience. In M. T. Cox & M. Freed (Eds.), *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms* (pp. 102-110). Menlo Park, CA: AAAI Press. (Available as Tech. Rep. No. SS-95-08)
- Oehlmann, R., Sleeman, D., & Edwards, P. (1993). Learning plan transformations from self-questions: A memory-based approach. In *Proceedings of the Eleventh National Conference on Artificial Intelligence* (pp. 520-525). Cambridge, MA: MIT Press.
- Osherson, D. N., Stob, M., & Weinstein, S. (1989). Learning theory and natural language. In R. M., & W. Demopoulos (Eds.), *Learnability and Linguistic Theory*. Boston: Kluwer Academic.
- Owens, C. (1990a). *Indexing and retrieving abstract planning knowledge*. Doctoral dissertation, Yale University, Department of Computer Science, New Haven, CT.
- Owens, C. (1990b). Representing abstract plan failures. In *Proceedings of Twelfth Annual Conference of the Cognitive Science Society* (pp. 277-284). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Owens, C. (1991). A functional taxonomy of abstract plan failures. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society* (pp. 167-172). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Owens, C. (1993). Integrating feature extraction and memory search. *Machine Learning*, 10 (3), 311-339.
- Palmer, S. E. (1978). Fundamental aspects of cognitive representation. In E. M. Rosch & B. B. Lloyd (Eds.), *Cognition and categorization* (pp. 259-303). Hillsdale, NJ: Lawrence Erlbaum Associates.

Park, Y. T., & Wilkins, D. C. (1990). Establishing the coherence of an explanation to improve refinement of an incomplete knowledge base. In *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 318-323). Menlo Park, CA: AAAI Press.

Patil, R. S., Fikes, R. E., Patel-Schneider, P. F., McKay, D., Finin, T., Gruber, T., & Neches, R. (1992). The DARPA knowledge sharing effort: Progress report. In B. N. C. Rich & W. Swartout (Eds.), *Principles of knowledge representation and reasoning* (pp. 777-788). San Francisco: Morgan Kaufmann.

Pazzani, M. (1990a). *Creating a memory of causal relationships: An integration of empirical and explanation-based learning methods*. Hillsdale, NJ: Lawrence Erlbaum.

Pazzani, M. (1990b). Learning fault diagnosis heuristics from device descriptions. In Y. Kodratoff & R. S. Michalski (Eds.), *Machine learning III: An artificial intelligence approach* (pp. 214-234). San Mateo, CA: Morgan Kaufmann.

Pazzani, M. (1991). Learning causal patterns: Deliberately overgeneralizing to facilitate transfer. In R. S. Michalski & G. Tecuci (Eds.), *Proceedings of the First International Workshop on Multistrategy Learning* (pp. 19-33). Washington, DC: George Mason University, Artificial Intelligence Center.

Pazzani, M. (1994). Learning causal patterns: Making a transition from data-driven to theory-driven learning. In R. Michalski & G. Tecuci (Eds.), *Machine learning IV: A multistrategy approach* (pp. 267-293). San Francisco: Morgan Kaufmann.

Pazzani, M. J., & Sarrett, W. (1990). Average case analysis of conjunctive learning algorithms. In B. W. Porter & R. J. Mooney (Eds.), *Machine Learning: Proceedings of the Seventh International Conference* (pp. 339-347). San Mateo, CA: Morgan Kaufmann.

Pirolli, P., & Bielaczyc, K. (1989). Empirical analyses of self-explanation and transfer in learning to program. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 450-457). Hillsdale, NJ: Lawrence Erlbaum Associates.

Pirolli, P., & Recker, M. (1994). Learning strategies and transfer in the domain of programming. *Cognition and Instruction*, 12(3), 235-275.

Plaza, E., Aamodt, A., Ram, A., van de Velde, W., & van Someren, M. (1992) Integrated learning architectures. In *Proceedings of the ECML-93 European Conference on Machine Learning*.

Plaza, E., & Arcos, J. L. (1993). Reflection and analogy in memory-based learning. In R. S. Michalski & G. Tecuci (Eds.), *Proceedings of the Second International Workshop on*

*Multistrategy Learning* (pp. 42-49). Fairfax, VA: George Mason University, Center for Artificial Intelligence.

Pollock, J. L. (1989a). *How to build a person*. Cambridge, MA: MIT Press/Bradford Books.

Pollock, J. L. (1989b). OSCAR: A general theory of rationality. *Journal of Experimental and Theoretical Artificial Intelligence*, 1, 209-226

Pope, A. (1713). *Windsor-forest: to the Right Honourable George Lord Lansdown/ by Mr. Pope.*, l. 11. London: Printed for Bernard Lintott.

Pressley, M., & Forrest-Pressley, D. (1985). Questions and children's cognitive processing. In A. C. Graesser & J. B. Black (Eds.), *The psychology of questions* (pp. 277-296). Hillsdale, NJ: Lawrence Erlbaum Associates.

Provost, F. J., & Buchanan, B. G. (1992). Inductive policy. In *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 255-261). Menlo Park, CA: AAAI Press.

Puerta, A., Egar, J., Tu, S., & Musen, M. (1992). A multiple-method knowledge-acquisition shell for the automatic generation of knowledge-acquisition tools. *Knowledge Acquisition*, 4, 171-196.

Punch, W. F., III (1991). TIPS (Task-Integrated Problem Solver), a task-specific integration architecture for heterogeneous agents. In *Proceedings of the AAAI Workshop on Cooperation Among Heterogeneous Intelligent Agents*. Anaheim, CA, (July, 15).

Punch, W. F., III, Goel, A. K., & Brown, D. C. (1996). A knowledge-based selection mechanism for strategic control with application in design, diagnosis and planning. *International Journal of Artificial Intelligence Tools*, 4(3), 323-348.

Pylyshyn, Z. W. (1991). The role of cognitive architecture in theories of cognition. In K. VanLehn (Ed.), *Architectures of cognition: The 22nd Carnegie Mellon symposium on cognition* (pp. 189-223). Hillsdale, NJ: Lawrence Erlbaum Associates.

Quilici, A. (in press). Toward automatic acquisition of an advisory system's knowledge base. *Applied Intelligence*.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1, 81-106.

Ram, A. (1989). *Question-driven understanding: An integrated theory of story understanding, memory and learning* (Tech. Rep. No. 710). Doctoral dissertation, Yale University, Department of Computer Science, New Haven, CT.

Ram, A. (1990a). Decision models: A theory of volitional explanation. In *Proceedings of Twelfth Annual Conference of the Cognitive Science Society* (pp. 198-205). Hillsdale, NJ: Lawrence Erlbaum Associates.

Ram, A. (1990b). Knowledge goals: A theory of interestingness. In *Proceedings of Twelfth Annual Conference of the Cognitive Science Society* (pp. 206-214). Hillsdale, NJ: Lawrence Erlbaum Associates.

Ram, A. (1991). A theory of questions and question asking. *Journal of the Learning Sciences*, 1, (3&4), 273-318.

Ram, A. (1993). Indexing, elaboration and refinement: Incremental learning of explanatory cases. *Machine Learning*, 10, 201-248.

Ram, A. (1994). AQUA: Questions that drive the understanding process. In R. C. Schank, A. Kass, & C. K. Riesbeck (Eds.), *Inside case-based explanation* (pp. 207-261). Hillsdale, NJ: Lawrence Erlbaum Associates

Ram, A., & Cox, M. T. (1994). Introspective reasoning using meta-explanations for multi-strategy learning. In R. S. Michalski & G. Tecuci (Eds.), *Machine learning IV: A multistrategy approach* (pp. 349-377). San Francisco: Morgan Kaufmann.

Ram, A., Cox, M. T., & Narayanan, S. (1992, July). An architecture for integrated introspective learning. In M. Weintraub (Ed.), *Proceedings of the ML-92 Workshop on Computational Architectures for Supporting Machine Learning & Knowledge Acquisition*, held at ML-92, Aberdeen, Scotland.

Ram, A., Cox, M. T., & Narayanan, S. (1995). Goal-driven learning in multistrategy reasoning and learning systems. In A. Ram & D. Leake (Eds.), *Goal-driven learning* (pp. 421-437). Cambridge, MA: MIT Press/Bradford Books.

Ram, A., & Hunter, L. (1992). The use of explicit goals for knowledge to guide inference and learning. *Applied Intelligence*, 2(1), 47-73.

Ram, A., & Jones, E. K. (1995). Foundations of *Foundations of artificial intelligence*. *Philosophical Psychology*, 8, 193-199.

Ram, A., & Leake, D. (1991). Evaluation of explanatory hypotheses. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society* (pp. 867-871). Hillsdale, NJ: Lawrence Erlbaum Associates.

Ram, A., & Leake, D. (1995). Learning, goals, and learning goals. In A. Ram & D. Leake (Eds.), *Goal-driven learning* (pp. 1-37). Cambridge, MA: MIT Press/Bradford Books.

- Ram, A., Narayanan, S., & Cox, M. T. (1995). Learning to trouble-shoot: Multistrategy learning of diagnostic knowledge for a real-world problem solving task. *Cognitive Science*, 19, 289-340.
- Reason, J. (1992). *Human error*. New York: Cambridge University Press.
- Recker, M., & Pirolli, P. (1995). Modeling individual differences in student's learning. *The Journal of the Learning Sciences*, 4(1), 1- 38.
- Reder, L. M. (1987). Strategy selection in question answering. *Cognitive Psychology*, 19, 90-138.
- Redmond, M. A. (1992). *Learning by observing and understanding expert problem solving* (Tech. Rep. No. GIT-CC-92/43). Doctoral dissertation, Georgia Institute of Technology, College of Computing, Atlanta.
- Reich, Y. (1994). Macro and micro perspectives of multistrategy learning. In R. S. Michalski & G. Tecuci (Eds.), *Machine learning IV: A multistrategy approach* (pp.379-401). San Francisco: Morgan Kaufmann.
- Reinders, M., & Bredeweg, B. (1992). Reflective strategic control of multiple problem solving methods. In B. Neumann (Ed.), *Proceedings of the 10th European Conference on Artificial Intelligence* (pp. 577-581). New York: John Wiley & Sons.
- Riesbeck, C. K., & Schank, R. C. (Eds.). (1989). *Inside case-based reasoning*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Rosenbloom, P. S., Laird, J. E., & Newell, A. (Eds.). (1993). *The Soar Papers: Research on integrated intelligence*. Cambridge, MA: MIT Press.
- Rosenfield, I. (1992). *The strange, familiar, and forgotten: An anatomy of consciousness*. New York: Alfred A. Knopf.
- Ross, B. H. (1989). Some psychological results on case-based reasoning. In *Proceeding of the DARPA Workshop on Case-Based Reasoning* (pp. 144-147). San Mateo, CA: Morgan kaufmann.
- Rumelhart, D. E., & McClelland, J. L. (Ed.). (1986). *Parallel distributed processing: Explorations in the microstructure of cognition* (Vols. 1 & 2). Cambridge, MA: MIT Press.
- Russell, S., & Wefald, E. (1991a). *Do the right thing: Studies in limited rationality*. Cambridge, MA: MIT Press.

Russell, S., & Wefald, E. (1991b). Principles of metareasoning. *Artificial Intelligence*, 49, 361-395.

Ryle, G. (1949). *The concept of mind*. New York: Barnes & Noble.

Sacerdoti, E. D. (1975). *A structure for plans and behavior* (Technical Note 109). Doctoral dissertation, SRI International, Inc., AI Center, Menlo Park, CA.

Sartre, J. P. (1965). *Situations I*, (B. Eisler, Trans.). New York: G. Braziller. (Original work published 1939)

Schaffer, C. (1993). Selecting a classification method by cross-validation. *Machine Learning*, 13(1), 135-143.

Schank, R. (1972). Conceptual dependency: A theory of natural language understanding. *Cognitive Psychology*, 3, 352-631.

Schank, R. (1975). *Fundamental studies in computer science, Vol. 3: Conceptual information processing*. Amsterdam: North-Holland Publishing.

Schank, R. (1979). Natural language, philosophy, and artificial intelligence. In M. Ringle (Ed.), *Philosophical perspectives in artificial intelligence* (pp. 196-224). Atlantic Highlands, NJ: Humanities Press.

Schank, R. C. (1982). *Dynamic memory: A theory of reminding and learning in computers and people*. Cambridge, MA: Cambridge University Press.

Schank, R. C. (1986). *Explanation patterns: Understanding mechanically and creatively*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Schank, R. C., & Abelson, R. P. (1977). *Scripts, plans, goals and understanding: An inquiry into human knowledge structures*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Schank, R. C., Collins, G. C., & Hunter, L. E. (1986). Transcending inductive category formation in learning. *Behavioral and Brain Sciences*, 9, 639-686.

Schank, R. C., Fano, A., Bell, B., & Jona, M. (1993). The design of goal-based scenarios. *The Journal of the Learning Sciences*, 3(4), 305-345.

Schank, R. C., Goldman, N., Rieger, C., & Riesbeck, C. K. (1972). *Primitive concepts underlying verbs of thought* (Stanford Artificial Intelligence Project Memo No. 162). Stanford, CA: Stanford University, Computer Science Department. (NTIS No. AD744634)



Schank, R. C., & Kass, A. (1990). Explanations, machine learning, and creativity. In Y. Kodratoff & R. S. Michalski (Eds.), *Machine learning III: An artificial intelligence approach* (pp. 31-48). San Mateo, CA: Morgan Kaufmann.

Schank, R. C., Kass, A., & Riesbeck, C. K. (1994). *Inside case-based explanation*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Schank, R. C., & Leake, D. (1990). Creativity and learning in a case-based explainer. In J. G. Carbonell (Ed.), *Machine learning: Paradigms and methods*. Cambridge, MA: MIT Press.

Schank, R. C., & Osgood, R. (1990). *A content theory of memory indexing*. Technical Report 2. Institute for the Learning Sciences, Northwestern University, Evanston, IL.

Schank, R. C., & Owens, C. C. (1987). Understanding by explaining expectation failures. In R. G. Reilly (Ed.), *Communication failure in dialogue and discourse*. New York: Elsevier Science.

Schank, R. C., & Riesbeck, C. (Eds.). (1981). *Inside computer understanding: Five programs plus miniatures*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Schank, R. C., & Tesler, L. G. (1969). A conceptual parser for natural language. In *Proceedings of the First International Joint Conference on Artificial Intelligence* (pp. 569-578). Washington, D. C.

Schneider, W. (1985). Developmental trends in the metamemory-memory behavior relationship: An integrative review. In D. L. Forrest-Pressley, G. E. MacKinnon, and T. G. Waller (Eds.), *Metacognition, cognition and human performance*. Vol. 1 (Theoretical perspectives; pp. 57-109). New York: Academic Press.

Schoenfeld, A. H. (1992). On paradigms and methods: What do you do when the ones you know don't do what you want them to? Issues in the analysis of data in the form of videotapes. *The Journal of the Learning Sciences*, 2(2), 179-214.

Schwanenflugel, P. J., Fabricius, W. V., Noyes, C. R., Bigler, K., D., & Alexander, J. M. (1994). The organization of mental verbs and folk theories of knowing. *Journal of Memory and Language*, 33, 376-395.

Searle, J. (1980). Minds, brains, and programs. *The Behavioral and Brain Sciences*, 3, 417-424.

Searle, J. (1990, July). Turing, Searle, & thought. *AI Expert*, 5, 52-59.

Searle, J. (1992). *The rediscovery of the mind*. Cambridge, MA: MIT Press/Bradford Books.

Self, J. (1992, August). BRM - A framework for addressing metacognitive issues in intelligent learning environments. In J. W. Brahan & G. E. Lasker (Eds.), *Proceedings of the Sixth International Conference on Systems Research, Informatics and Cybernetics: Vol. 2. Advances in Artificial Intelligence - Theory and Application* (pp. 85-90). Windsor, Ontario, Canada: International Institute for Advanced Studies in Systems Research and Cybernetics.

Siegler, R. S. (1988). Individual differences in strategy choices: Good students, not-so-good students, and perfectionists. *Child Development*, 59, 833-852.

Siegler, R. S. (1991). *Children's thinking* (2e). Englewood Cliffs, NJ: Prentice Hall.

Simina, M. D., & Kolodner, J. L. (1995). Opportunistic reasoning: A design perspective. In J. D. Moore & J. F. Lehman (Eds.), *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society* (pp. 78-83). Hillsdale, NJ: Lawrence Erlbaum Associates.

Simon, H. A. (1979). What the knower knows: Alternative strategies for problem-solving tasks. In F. Klix (Ed.), *Human and artificial intelligence* (pp. 89-100). Amsterdam: North Holland.

Simon, H. A. (1983). Why should machines learn? In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning I: An artificial intelligence approach* (pp. 25-37). Los Altos, CA: Morgan Kaufmann.

Simon, H. A. (1995). Artificial intelligence: an empirical science. *Artificial Intelligence*, 77, 95-127.

Simon, T. J., & Halford, G. S. (1995). Computational models and cognitive change. In T. Simon & G. Halford (Eds.), *Developing cognitive competence* (pp. 1-30). Hillsdale, NJ: Lawrence Erlbaum Associates.

Simpson, R. L. (1985). *A computer model of case-based reasoning in problem solving: An investigation in the domain of dispute mediation* (Tech. Rep. No. GIT-ICS-85/18). Doctoral dissertation, Georgia Institute of Technology, College of Computing, Atlanta.

Skinner, B. F. (1950). Are theories of learning necessary? *Psychological Review*, 57, 193-216.

Skinner, B. F. (1956). What is psychotic behavior? In F. Gildea (Ed.), *Theory and treatment of the psychoses: Some newer aspects*. St. Louis: Washington University Press.

- Sleeman, D., Langley, P., & Mitchell, T. M. (1984). *Learning from solution paths: An approach to the credit assignment problem*. CIP Working Paper 443. Carnegie Melon University. Pittsburgh, PA.
- Smith, B. C. (1985). Prologue to "Reflection and semantics in a procedural language". In R. J. Brachman & H. J. Levesque (Eds.), *Readings in Knowledge Representation* (pp. 31-40). San Mateo, CA: Morgan Kaufmann. (Original work published 1982)
- Smyth, B., & Keane, M. T. (1995). Remembering to forget: A competence-preserving case deletion policy for case-base reasoning systems. In C. S. Mellish (Ed.), *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* (pp. 377- 382). San Mateo, CA: Morgan Kaufmann.
- Spear, N. E. (1978). *The Processing of Memories: Forgetting and retention*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Stallman, R. M., & Sussman, G. J. (1977). Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence*, 9, 135-196.
- Steels, L. (1990). Components of expertise. *AI Magazine*, 11(2), 30-49.
- Stefik, M. (1981). Planning and metapanning (MOLGEN: Part 2). *Artificial Intelligence*, 16, 141-169.
- Stein, G. & Barnden, J. A. (1995). Towards more flexible and common-sensical reasoning about beliefs. In M. T. Cox & M. Freed (Eds.), *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms* (pp. 127-135). Menlo Park, CA: AAAI Press. (Available as Tech. Rep. No. SS-95-08)
- Steier, D. M., Laird, J. E., Newell, A., Rosenbloom, P. S., Flynn, R., Golding, A., Polk, T. A., Shivers, O. G., Unruh, A., & Yost, G. R. (1993). Varieties of learning in Soar: 1987. In P. S. Rosenbloom, J. E. Laird, & A. Newell (Eds.), *The Soar Papers: Research on integrated intelligence* (Vol. 1, 537- 548). Cambridge, MA: MIT Press. (Original work published 1987)
- Stevens, W. (1954). To an old philosopher in Rome [Stanza 16]. *Collected poems of Wallace Stevens* (pp. 508-511). New York: Alfred A. Knopf. (Original work published 1952)
- Stich, S. P., & Warfield, T. A. (Eds.). (1994). *Mental representation: A reader*. Cambridge, MA: Blackwell.

Stroulia, E. (1994). *Failure-driven learning as model-based self-redesign*. Doctoral dissertation, Georgia Institute of Technology, College of Computing, Atlanta.

Stroulia, E., & Goel, A. (1992). A model-based approach to incremental self-adaptation. In M. Weintraub (Ed.), *Proceedings of the ML-92 Workshop on Computational Architectures for Supporting Machine Learning & Knowledge Acquisition*. Aberdeen, Scotland, (July, 4).

Stroulia, E., & Goel, A. (1995). Functional representation and reasoning for reflective systems. *Journal of Applied Intelligence*, 9(1), 101-124.

Stroulia, E., Shankar, M., Goel, A., & Penberthy, L. (1992). A model-based approach to blame assignment in design. In J. S. Gero (Ed.), *Proceedings of AID'92: Second International Conference on AI in Design* (pp. 519-537).

Suchman, L. (1987). *Plans and Situated Action: The problem of human-machine communication*. Cambridge, MA: Cambridge Press.

Sussman, G. J. (1975). *A computer model of skill acquisition*. New York: American Elsevier.

Sutton, R. S. (Ed.). (1992). Reinforcement learning [Special issue]. *Machine Learning*, 8 (3/4).

Swanson, H. L. (1990). Influence of metacognitive knowledge and aptitude on problem solving. *Journal of Educational Psychology*, 82(2), 306-314.

Sycara, E. P. (1987). *Resolving adversarial conflicts: An approach to integrating case-based and analytic methods* (Tech. Rep. No. GIT-ICS-87/26). Doctoral dissertation, Georgia Institute of Technology, School of Information and Computer Science, Atlanta.

Tash, J. & Russell, S. (1994). Control strategies for a stochastic planner. In *Proceedings of the Twelfth National Conference on Artificial Intelligence, II* (pp. 1079-1085). Cambridge, MA: MIT Press.

Tate, A. (1976). *Project planning using a hierarchic non-linear planner* (Tech. Rep. No. 25). Edinburgh, UK: University of Edinburgh, Department of Artificial Intelligence.

Tate, A. (1990). Generating project networks. In J. Allen, J. Hendler, & A. Tate (Eds.), *Readings in planning* (pp. 291-296). San Mateo, CA: Morgan Kaufmann. (Original work published 1977)

Thagard, P. (1992). Adversarial problem solving: Modeling an opponent using explanatory coherence. *Cognitive Science*, 16, 123-149.

- Thagard, P. (1993). *Computational philosophy of science*. Cambridge, MA: MIT Press/Bradford Books. (Original work published 1988)
- Titchener, E. B. (1912). The schema of introspection. *The American Journal of Psychology*, 23(4), 485-508.
- Tulving, E. (1994). Forward. In J. Metcalfe & A. P. Shimamura (Eds.), *Metacognition: Knowing about knowing* (pp. vii-xiii). Cambridge, MA: MIT Press/Bradford Books.
- Turing, A. M. (1963). Computing machinery and intelligence. In E. A. Feigenbaum & J. Feldman (Eds.), *Computers and thought* (pp. 11-35). New York: McGraw Hill. (Original work published 1950)
- Turner, R. (1989). *A schema-based model of adaptive problem solving* (Tech. Rep. No. GIT-ICS-89/42). Doctoral dissertation, Georgia Institute of Technology, School of Information and Computer Science, Atlanta.
- VanLehn, K. (Ed.). (1991a). *Architectures for intelligence: The 22nd Carnegie Mellon symposium on cognition*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- VanLehn, K. (1991b). Rule acquisition events in the discovery of problem solving strategies. *Cognitive Science*, 15, 1-47.
- VanLehn, K., Jones, R. M., & Chi, M. T. H. (1992). A model of the self-explanation effect. *Journal of the Learning Sciences*, 2(1), 1-60.
- Veloso, M. (1994). *Planning and learning by analogical reasoning*. Springer-Verlag.
- Veloso, M., & Carbonell, J. G. (1990). Integrating analogy into a general problem-solving architecture. In M. Zemankova & Z. Ras (Eds.), *Intelligent systems*. Chichester, UK: Ellis Horwood.
- Veloso, M., & Carbonell, J. G. (1994). Case-based reasoning in PRODIGY. In R. S. Michalski & G. Tecuci (Eds.), *Machine learning IV: A multistrategy approach* (pp.523-548). San Francisco: Morgan Kaufmann.
- Watson, J. B. (1919). *Psychology from the standpoint of the behaviorist*. Philadelphia: J. B. Lippincott.
- Weinert, F. E. (1987). Introduction and overview: Metacognition and motivation as determinants of effective learning and understanding. In F. E. Weinert & R. H. Kluwe (Eds.), *Metacognition, motivation, and understanding* (pp. 1-16). Hillsdale, NJ: Lawrence Erlbaum Associates.

Weintraub, M. A. (1991). *An explanation-based approach to assigning credit*. Doctoral dissertation, Ohio State University, Columbus.

Weld, D., & Etzioni, O. (1994). The first law of robotics (a call to arms). In *Proceedings of the Twelfth National Conference on Artificial Intelligence* (Vol. 2, pp. 1042-1047). Menlo park, CA: AAAI Press.

Wellman, H. M. (1983). Metamemory revisited. In M. T. H. Chi (Ed.), *Contributions to Human Development*. Vol. 9 (Trends in memory development research). S. Karger, AG, Basel, Switzerland.

Wellman, H. M. (1985). The origins of metacognition. In D. L. Forrest-Pressley, G. E. MacKinnon, and T. G. Waller (Eds.), *Metacognition, cognition and human performance*. Vol. 1 (Theoretical perspectives, pp. 1-31). New York: Academic Press.

Wellman, H. M. (1992). *The child's theory of mind*. Cambridge, MA: MIT Press.

Wellman, H. M., & Johnson, C. N. (1979). Understanding of mental process: A developmental study of "remember" and "forget." *Child Development*, 50, 79-88.

Wexler, K., & Cullicover, P. (1980). *Formal principles of language acquisition*. Cambridge, MA: MIT Press.

Wilensky, R. (1978). *Understanding goal-based stories* (Tech. Rep. No. 140). Doctoral dissertation, Yale University, Department of Computer Science, New Haven, CT.

Wilensky, R. (1983). *Planning and understanding: A computational approach to human reasoning*. Reading, MA: Addison-Wesley Publishing.

Wilensky, R. (1986a). Knowledge representation: A critique and a proposal. In J. L. Kolodner & C. K. Riesbeck (Eds.), *Experience, memory, and reasoning* (pp. 15-28). Hillsdale, NJ: Lawrence Erlbaum Associates.

Wilensky, R. (1986b). *Some problems and proposals for knowledge representation* (Tech. Rep. No. UCB/CSD 86/294). Berkeley, CA: University of California.

Wilson, T. D., & Schooler, J. W. (1991). Thinking too much: Introspection can reduce the quality of preferences and decisions. *Journal of Personality and Social Psychology*, 60(2), 181-192.

Winograd, T. (1985). Frame representations and the declarative/procedural controversy. In R. J. Brachman & H. J. Levesque (Eds.), *Readings in Knowledge Representation* (pp. 358-370). San Mateo, CA: Morgan Kaufmann. (Original work published 1975)

Winston, P. H., Binford, T. O., Katz, B., & Lowry, M. (1983). Learning physical descriptions from functional definitions, examples, and precedents. In *Proceedings of the National Conference on Artificial Intelligence* (pp. 433-439). San Francisco: Morgan Kaufmann.

Wisniewski, E. J., & Medin, D. L. (1991). Harpoons and long sticks: The interaction of theory and similarity in rule induction. In D. H. Fisher, M. J. Pazzani, & P. Langley (Eds.), *Concept formation: Knowledge and experience in unsupervised learning* (pp. 237-278). San Mateo, CA: Morgan Kaufmann.

Yussen, S. R. (Ed.). (1985). *The growth of reflection in children*. New York: Academic Press.

Zeichick, A. L. (1992, January). The Turing test. *AI Expert*.





## NAME INDEX

### A

Aamodt, A. 152, 299  
 Abelson, R. P. 26, 146, 182, 295, 296  
 Agnew, N. 54  
 Aho, A. 30  
 Alexander, J. M. 69  
 Ali, K. S. 107  
 Allen, J. 118, 247, 248, 269  
 Almonayyes, A. 75  
 Anderson, J. R. 117, 230, 274  
 Antaki, C. 273  
 Arcos, J. L. 173  
 Arkin, R. 54  
 Ashley, K. D. 134  
 Asimov, I. 293  
 Augustine 263  
 Ayer, A. J. 211

### B

Bach, J. S. 263  
 Bain, W. M. 134, 135  
 Baird, W. B. vi  
 Balch, T. vi  
 Ballim, A. 266  
 Banerji, R. 54, 201, 313  
 Barnden, J. 67, 265  
 Barr, A. 267  
 Barsalou, L. W. vi, ix, x, 13, 117, 257, 258, 260  
 Bass, E. vi  
 Bassok, M. 220, 258  
 Batali, J. 268  
 Bell, B. 258

Bereiter, C. 151  
 Bhatta, S. vi, 192, 265  
 Bielaczyc, K. 278  
 Bigler, K. D. 69  
 Billington, R. vi  
 Binford, T. O. 55  
 Birnbaum, L. 25, 31, 32, 41, 60, 122, 155, 173, 270, 271, 294  
 Bock, C. 267  
 Booker, L. B. 201  
 Boring, E. G. 8, 242  
 Bransford, J. D. 265  
 Brazdil, P. B. 264  
 Bredeweg, B. 107  
 Brewer, W. F. 255  
 Brigham, M. C. 276  
 Brodley, C. 173, 197, 248  
 Brown, A. 108, 241, 265, 272, 280  
 Brown, L. 283  
 Buchanan, B. G. 173, 267, 268, 305, 313  
 Byrne, M. vi

### C

Callantine, T. J. 107  
 Carbonell, J. G. 10, 18, 84, 117, 118, 119, 134, 177, 211, 251, 268, 269, 270  
 Cardie, C. 220  
 Carey, S. 153  
 Carpenter, S. vi  
 Ceci, S. J. 272  
 Celan, P. 3  
 Chalasani, P. 118, 248, 269  
 Chalmers, D. 293  
 Chandrasekaran, B. 107

Cheeseman, P. 196  
 Cheng, J. 118, 248, 269, 271  
 Chi, M. T. H. 220, 258, 278, 279, 301, 302, 305  
 Chinn, C. A. 255  
 Clancey, W. J. 54, 267, 268, 305  
 Cohen, S. M. 238  
 Coleridge, S. T. 133  
 Collins, G. C. 31, 41, 60, 122, 155, 173, 174, 270, 271  
 Converse, T. 60, 147, 160, 311  
 Cox, M. T. i, ix, 7, 8, 9, 11, 12, 16, 17, 18, 19, 20, 21, 22, 23, 27, 28, 41, 52, 74, 76, 79, 93, 114, 122, 123, 124, 147, 213, 238, 240, 255, 260, 264, 270, 278, 299, 300  
 Cox, B. M. iv  
 Cox, C. N. iv  
 Cox, J. S. W. vii  
 Cox, R. M. vi  
 Crockett, L. J. 304  
 Cullicover, P. 153  
 Cullingford, R. 26, 182, 220

## D

Danyluk, A. P. 118  
 Davidson, H. 275, 279, 280  
 Davis, R. 18, 267, 268, 305  
 DeJong, G. 162  
 deKleer, J. 265, 270  
 Delclos, V. R. 274, 280  
 Dennett, D. C. 266, 293  
 Derry, S. J. 274, 302  
 Descartes, R. 67, 69, 304, 305, 306  
 desJardins, M. 311  
 Deuser, R. 275  
 Devaney, M. vi  
 Dixon, R. A. 275  
 Domeshek, E. A. vi, 25, 51  
 Donnellan, M. W. 107  
 Dörner, D. 273, 274  
 Dostoyevsky, F. 42

Doyle, J. 18, 19, 72, 76, 77, 243, 266, 267, 269, 270, 305  
 Draper, D. 170, 296  
 Dunbar, K. 110  
 Dunlosky, J. vi, 50, 279, 280  
 Dyer, M. G. 220

## E

Edelman, G. M. 293  
 Edwards, P. 90, 107, 139, 173, 206, 270  
 Egar, J. 108  
 Eiselt, K. iii, vi  
 Epstein, W. 72, 242  
 Escher, M. C. 263  
 Etzioni, O. 118, 170, 248, 269, 293, 296  
 Evens, M. 260

## F

Fabricius, W. V. 69  
 Fano, A. 258  
 Farrell, S. J. vi  
 Fikes, R. E. 21, 32, 160  
 Finin, T. 32  
 Fischhoff, B. 8  
 Fisher, D. H. 220  
 Fixx, J. 156  
 Flavell, J. H. 272, 277  
 Foote, B. vi  
 Ford, K. M. 54  
 Forrest-Pressley, D. L. 220, 272, 277  
 Fox, S. 41, 69, 173, 270, 271  
 Freed, M. vi, 22, 41, 122, 173, 174, 264, 270, 271  
 Freeman, D. 196  
 Freeman, P. vi

## G

Gardner, H. 54  
 Garner, R. 272, 277

Garza, A. G. 107  
 Gavelek, J. R. 220, 277  
 Genesereth, M. R. 267, 268, 305  
 Ghosh, S. 22, 127, 159, 178, 197, 385  
 Gil, Y. 251  
 Glasser, R. 220, 258  
 Glenberg, A. M. 72, 242  
 Goda, J. vi  
 Gödel, K. 263  
 Goel, A. K. 107, 108, 122, 173, 265, 270, 271  
 Goldberg, D. E. 201  
 Goldman, N. 18, 74, 101, 152  
 Goldstein, I. 267, 305  
 Gombert, J. E. 277  
 Govindaraj, T. 238  
 Green, C. C. 153  
 Greeno, J. G. 295  
 Gruber, T. R. 32

## H

Hale, C. R. vi, ix, x, 258, 260  
 Halford, G. S. 212  
 Hammond, K. J. 17, 41, 42, 44, 60, 128, 134, 147, 155, 156, 160, 270, 311  
 Handey, J. 153, 293  
 Hanks, S. 170, 296  
 Harley, S. 220  
 Harrington, C. 274, 280  
 Hayes, P. J. 30, 31, 54, 72, 73, 265  
 Hayes-Roth, B. 60  
 Hayes-Roth, F. 41, 60, 79  
 Heinlein, R. A. 293  
 Hendler, J. 22, 127, 159, 178, 197, 247, 385  
 Hertzog, C. K. vi, 275  
 Hinrichs, T. R. vi, 134  
 Hmelo, C. vi  
 Hofstadter, D. R. 263, 293  
 Holland, J. H. 201  
 Horthy, J. 264  
 Hultsch, D. F. 275  
 Hume, G. 260

Hunter, L. E. 9, 10, 12, 20, 31, 93, 114, 118, 122, 123, 147, 152, 155, 158, 167, 173

## J

Jameson, A. 254  
 Johnson, H. M. 63, 201, 220, 278  
 Johnson, J. S. iv  
 Johnson-Laird, P. N. 265  
 Jona, M. 258  
 Jones, E. K. 191, 299, 306  
 Jones, R. M. 41, 278

## K

Kambhampati, S. 22, 127, 159, 178, 197, 385  
 Kass, A. 20, 42, 75, 134, 137, 156, 270  
 Katz, B. 55  
 Kausler, D. H. 275, 276  
 Keane, M. T. 78  
 Kedar-Cabelli, S. 55, 162, 270  
 Keil, F. C. 153  
 Kell, A. 22  
 Keller, R. M. 55, 122, 162, 270  
 Kelly, J. 196  
 Kennedy, A. C. 69, 270  
 Kerner, Y. 75, 152  
 Kettler, B. 22, 127, 159, 178, 197, 385  
 Klahr, D. 110  
 Kluwe, R. H. 274  
 Knoblock, C. A. 117, 118, 119, 269  
 Kolodner, J. L. iii, v, vi, 41, 51, 60, 130, 134, 135, 152, 177, 192  
 Konolige, K. 264, 266  
 Koton, P. 134  
 Krinsky, R. 50  
 Krulwich, B. 41, 90, 122, 173, 270, 271  
 Kuokka, D. R. 79, 107, 118, 269

## L

Lachman, J. L. 278  
 Lachman, R. 278  
 Laird, J. E. 13, 117, 269  
 Langley, P. 201  
 Leake, D. B. vi, 10, 12, 20, 41, 42, 69, 75,  
 122, 134, 137, 147, 173, 211, 259,  
 270, 271, 273  
 Lebowitz, M. 165  
 Lee, M. S. 248  
 Lehnert, W. 220  
 Lenat, D. B. 79, 267, 305  
 Leonesio, R. J. 254  
 Lesh, N. 170, 296  
 Lewis, A. 220, 258, 273  
 Lichtenstein, S. 8  
 Lieberman, D. A. 242  
 Loebner, H. G. 302  
 Lovelace, E. A. 275, 276  
 Lowry, M. 55  
 Lyons, W. 263

## M

MacKinnon, G. E. 272  
 Maes, P. 264, 268, 269  
 Mahesh, K. vi  
 Markovitch, S. 78, 177, 209  
 Marks, M. 60, 147, 311  
 Marsh, G. R. 275  
 Martin, C. E. 134, 177, 207  
 Martin, J. vi, 251  
 McCarthy, J. 30, 31, 72, 73, 220, 264, 265,  
 266, 267, 293  
 McClelland, J. L. 123  
 McDermott, D. 77, 267  
 McDermott, J. 107  
 McKay, D. 32  
 McLaren, B. M. 134  
 McNamara, T. P. 265  
 McRoy, S. W. 72  
 Medin, D. L. 117, 251

Meehan, J. 22, 178, 184, 188  
 Metcalfe, J. 290  
 Michael, J. 260  
 Michalski, R. S. 8, 12, 116, 117, 122, 165,  
 248  
 Miller, D. L. 265  
 Miner, A. C. 278, 279  
 Minsky, M. L. 122, 123, 178, 264, 265,  
 299, 302  
 Minton, S. 17, 61, 117, 118, 119, 269, 270,  
 311  
 Mitchell, T. M. 54, 55, 118, 123, 153, 162,  
 201, 238, 269, 270, 313  
 Mooney, R. 56, 162, 256, 270, 271  
 Moore, A. W. 248  
 Moore, R. C. 31, 266  
 Moorman, K. vi, 153  
 Morik, K. 118  
 Musen, M. 108

## N

Narayanan, S. vi, 17, 23, 238, 240, 260, 299  
 Nardi, D. 264  
 Narens, L. 241, 254, 269, 276, 277  
 Neches, R. 32  
 Neisser, U. 305  
 Nelson, T. O. 50, 241, 254, 269, 276, 277,  
 279, 280  
 Nersessian, N. J. iii, vi, 13  
 Newell, A. 13, 17, 25, 30, 47, 54, 107, 117,  
 124, 146, 195, 212, 268, 269, 294,  
 296, 299  
 Ng, E. 151  
 Nilsson, N. J. 21, 160  
 Nisbett, R. E. 242  
 Nolan, E. vii  
 Noyes, C. R. 69

## O

Oehlmann, R. vi, 90, 107, 139, 173, 206,  
 270

Osgood, R. 51  
 Osherson, D. N. 153  
 Ourston, D. 56, 256, 270, 271  
 Owens, C. 20, 41, 42, 49, 51, 59, 75, 128,  
 134, 137, 154, 155, 156, 177, 192,  
 270, 271

## P

Palmer, S. E. 31  
 Park, Y. T. 178, 270, 271  
 Patel-Schneider, P. F. 32  
 Patil, R. S. 32  
 Pazzani, M. J. vi, 41, 118, 134, 185, 212,  
 270  
 Pearce, M. vi  
 Penberthy, L. 122  
 Peterson, J. vi  
 Pirollo, P. 4, 22, 151, 218, 229, 230, 231,  
 233, 253, 258, 278  
 Plaza, E. 173, 299  
 Pollock, J. L. 263, 273, 299  
 Pope, A. vii, 263  
 Pressley, M. 220, 276, 277  
 Provost, F. J. vi, 173, 313  
 Puerta, A. 108  
 Punch, W. F., III 107, 108  
 Pylyshyn, Z. W. 117, 123, 153

## Q

Quilici, A. 173  
 Quinlan, J. R. 248, 311

## R

Ram, A. iii, v, vi, 7, 8, 9, 10, 11, 12, 16, 17,  
 18, 19, 20, 21, 23, 27, 41, 42, 45, 52,  
 60, 74, 75, 77, 80, 81, 82, 93, 110,  
 111, 114, 116, 122, 123, 124, 134,  
 137, 147, 152, 153, 167, 173, 178,  
 182, 191, 195, 211, 221, 238, 240,

248, 255, 260, 273, 278, 296, 299,  
 306

Ram Das 283  
 Raphael, T. E. 220, 277  
 Reason, J. 41, 61  
 Recker, M. iii, vi, 4, 22, 151, 229, 230, 231,  
 233, 253, 258, 278  
 Reder, L. M. 61, 278, 279  
 Redmond, M. A. vi, 134, 136, 156, 158,  
 173, 270  
 Reich, Y. 118  
 Reimann, P. 220, 258  
 Reinders, M. 107  
 Reiser, B. J. 230  
 Rieger, C. 18, 74, 101, 152  
 Riesbeck, C. K. 18, 74, 75, 101, 134, 152,  
 220  
 Riley, M. S. 295  
 Riloff, E. 220  
 Ringuette, M. 118, 248, 269  
 Rissland, E. L. 134  
 Roberts, J. vi  
 Rosenbloom, P. S. 13, 117, 269  
 Rosenfield, I. 293  
 Ross, B. H. 135  
 Rovick, A. 260  
 Rumelhart, D. E. 123  
 Rumiano, E. vi  
 Russell, C. vi  
 Russell, S. 269  
 Ryle, G. 31

## S

Sacerdoti, E. D. 159  
 Salthouse, T. vi  
 Santamaria, J. C. vi  
 Sarrett, W. 212  
 Sartre, J. P. 245  
 Schaffer, C. 173, 248  
 Schank, R. C. 18, 20, 26, 31, 41, 51, 68, 69,  
 71, 73, 74, 75, 101, 134, 137, 146,  
 152, 155, 177, 182, 220, 249, 258,

259, 295, 296, 303, 304, 305, 306  
 Schlimmer, J. C. 118, 248, 269  
 Schneider, W. 272, 277, 279  
 Schoenfeld, A. H. 212  
 Schooler, J. W. 8, 242  
 Schrobe, H. 267, 305  
 Schwanenflugel, P. J. 69  
 Scott, P. D. 78, 177, 209  
 Searle, J. 285, 289, 303, 305, 306  
 Seifert, C. M. 60, 63, 147, 201, 311  
 Self, J. 268, 276  
 Self, M. 196  
 Shankar, M. 122  
 Shimamura, A. P. 290  
 Shoham, Y. 264  
 Siegler, R. S. 41, 61  
 Siegmund, P. vi  
 Simina, M. 60  
 Simon, H. A. 54, 107, 115, 116, 124, 146, 212, 294  
 Simon, T. iii, vi, 212  
 Simpson, R. L. 134, 152  
 Skinner, B. F. 242  
 Sleeman, D. 90, 107, 139, 173, 201, 206, 270  
 Slovic, P. 8  
 Smith, A. D. vi  
 Smith, B. C. 268  
 Smith, R. G. 305, 378  
 Smyth, B. 78  
 Soderland, S. 220  
 Spear, N. E. 278  
 Stallman, R. M. 72  
 Steele, G. L. 270  
 Steels, L. 107  
 Stefik, M. 108, 268  
 Steier, D. M. 117  
 Stein, G. 67, 265  
 Sternberg, R. J. 275  
 Stevens, W. 177  
 Stich, S. P. 267  
 Stob, M. 153  
 Stroulia, E. vi, 41, 122, 173, 201, 270, 271

Stutz, J. 196  
 Suchman, L. 54  
 Sundheim, B. 220  
 Sussman, G. J. 12, 41, 72, 157, 247, 270, 288  
 Sutton, R. S. 123  
 Swanson, H. L. 274, 280  
 Sycara, E. P. 134

## T

Tash, J. 269  
 Tate, A. 22, 127, 159, 160, 247, 249, 290  
 Taylor, W. 196  
 Tecuci, G. 8, 117  
 Tesler, L. G. 71, 101  
 Thagard, P. 13, 312  
 Thompson, R. 274  
 Thronesbery, C. 278  
 Titchener, E. B. 242  
 Tu, S. 108  
 Tulving, E. 31  
 Turing, A. 47, 265, 294, 302, 303, 304, 305, 306  
 Turner, R. 134

## U

Ullman, J. D. 30  
 Utgoff, P. E. 54, 201, 313

## V

van de Velde, W. 299  
 van Someren, M. 299  
 VanLehn, K. 41, 278, 299  
 Veloso, M. 18, 84, 118, 119, 134, 177, 270

## W

Waller, T. G. 272  
 Warfield, T. A. 267

Waterman, D. A. 79  
Watson, J. B. 242  
Wefald, E. 269  
Weinert, F. E. 241, 279  
Weinstein, S. 153  
Weintraub, M. A. 122  
Weld, D. 170, 293, 296  
Wellman, H. M. 272, 273, 277, 278, 279,  
299, 301, 302, 305  
Wexler, K. 153  
Wilensky, R. 45, 71, 81, 111, 178, 191, 220,  
247, 294  
Wilkins, D. C. 178, 270, 271  
Wilkinson, A. C. 72, 242  
Williamson, M. 170, 296  
Wilson, T. D. 8, 242  
Winograd, T. 67  
Winston, P. H. 55  
Wisniewski, E. J. 117, 251  
Wolff, J. S. vii

## Y

Yang, C. 220  
Yussen, S. R. 272

## Z

Zeichick, A. L. 302  
Ziolko, A. vi





## PROGRAM INDEX

### A

ACT\* 274  
 ANON 155, 177  
 AQ3 153  
 AQUA 20, 21, 22, 111, 137, 138, 178, 180,  
     186, 233  
 AUTOCLASS 196  
 Autognostic 201, 271, 272

### B

BORIS 220  
 BRIDGER 118

### C

CASTLE 271, 272  
 CELIA 136  
 CHEF 156  
 CMU LISP tutor 230  
 CYRUS 177

### D

DMAP 177, 207

### E

EITHER 256, 271

### F

FUNES 177

### G

GEMINI 118

### I

ID3 248, 311  
 INVESTIGATOR 118, 152  
 ISM 118, 248, 271  
 IULIAN 107, 270

### L

LEX 54, 313

### M

MAX 107  
 MECH ix, x  
 Meta-AQUA vi, xxix, 3, 10, 11, 12, 20, 21,  
     22, 23, 26, 27, 28, 29, 31, 32, 45, 56,  
     59, 62, 65, 77, 78, 87, 102, 110, 111,  
     112, 118, 133, 134, 136, 137, 138,  
     139, 140, 143, 149, 151, 152, 154,  
     155, 156, 158, 167, 168, 170, 171,  
     173, 177, 178, 179, 180, 181, 182,  
     184, 185, 186, 188, 190, 192, 194,  
     195, 196, 198, 200, 201, 202, 203,  
     204, 206, 207, 209, 212, 213, 214,  
     215, 218, 219, 221, 224, 225, 227,  
     229, 230, 231, 233, 234, 236, 241,  
     242, 243, 246, 248, 249, 250, 251,  
     253, 256, 257, 263, 269, 270, 271,  
     272, 278, 279, 281, 285, 288, 289,  
     294, 295, 303, 311

Meta-TS 23, 230, 238, 239, 241, 289, 294  
 MINERVA 178, 271  
 MOBAL 118  
 MOLGEN 108

## N

NOAH 159  
 Nonlin 22, 127, 159, 167, 168, 178, 181,  
 197, 249

## O

OCCAM 118, 185, 270

## P

PAGODA 311, 312  
 PRODIGY 18, 61, 118, 119, 269, 311, 378  
 PROTÉGÉ-II 108  
 PUPS 274

## R

RAPTER 270, 272  
 REFLECT 107  
 RFermi 270  
 ROBBIE 271, 272  
 ROUTER 107

## S

SAM 182  
 SMART 177  
 Soar 13, 47, 96, 117, 218, 269, 390  
 STRIPS 21, 160, 197  
 SURF 218  
 SWALE 20, 137, 138, 156

## T

Tale-Spin vi, 22, 178, 179, 181, 184, 185,  
 186, 188, 190, 194, 196, 201, 203,  
 209, 219, 221, 222, 243, 252, 289  
 TEIRESIAS 268  
 Theo 118, 248, 269  
 TIPS 107  
 TWEAKER 156

## U

UNIMEM 165

## SUBJECT INDEX

### A

Aamodt, A. 152, 299  
 abduction 117, 123  
 Abelson, R. P. 26, 146, 182, 295, 296  
 abstract cases 152  
 abstraction 118, 167, 197, 202, 248  
 abstraction schema 167  
 abstraction transmutation 168  
 achievement goal 146  
 ACT\* 274  
 action schema 165  
 action schemas 159  
 active failure generation 209  
 actschemas 159  
 actual outcome 44, 45, 48, 49, 95, 96, 120, 312  
*ad hoc* categories 257  
 affect 68  
 Agnew, N. 54  
 Aho, A. 30  
 AI v, 18, 198, 212, 264, 290, 298  
 Alexander, J. M. 69  
 Ali, K. S. 107  
 Allen, J. 118, 247, 248, 269  
 Almonayyes, A. 75  
 analogical learning 118  
 analogy 18, 84, 114, 117, 118, 124, 255  
 Anderson, J. R. 117, 230, 274  
 Andrew (cartoon character) 5, 6, 7, 16, 18, 42, 47  
 annotations 35  
 anomaly 26, 27, 35, 45, 84, 108, 109, 112, 114, 130, 136, 200, 201  
 anomaly identification 33, 83, 113, 120  
 ANON 155, 177

Antaki, C. 273  
 anticipate-and-avoid 44  
 applicability conditions 164  
 apprehension 254, 255  
 appropriateness condition 160  
 AQ3 153  
 AQUA 20, 21, 22, 111, 137, 138, 178, 180, 186, 233  
 Arcos, J. L. 173  
 Arkin, R. 54  
 ARPA 32  
 artificial intelligence (see AI) 246, 263  
 Ashley, K. D. 134  
 Asimov, I. 293  
 associative matrix technique 155  
 assumptions 105, 106, 107, 186, 271  
 ATRANS 71, 182  
 attend 255  
 attentional mechanisms 63  
 attribute value 191  
 Augustine 263  
 AUTOCLASS 196  
 Autognostic 201, 271, 272  
 automata 304  
 automatic programming systems 305  
 automatic theorem prover 8  
 average-case mathematical model 212  
 Ayer, A. J. 211

### B

Bach, J. S. 263  
 background knowledge (see BK) 7, 11, 19, 26, 73, 76, 106, 107, 110, 122, 123, 124, 129, 130, 137, 140, 149, 152,

- 157, 178, 255, 278, 284, 288  
 baffled 255  
 Bain, W. M. 134, 135  
 Baird, W. B. vi  
 Balch, T. vi  
 Ballim, A. 266  
 Banerji, R. 54, 201, 313  
 Barnden, J. 67, 265  
 Barr, A. 267  
 Barsalou, L. W. vi, ix, x, 13, 117, 151, 257, 258, 260  
 Base IMXP 84, 87  
 basic level categories 90  
 Bass, E. vi  
 Bassok, M. 220, 258  
 Batali, J. 268  
 Behaviorism 242  
 belated prediction 87, 90, 98  
 belief 266, 269, 273, 279  
 belief revision 72  
 belief-desire psychology 301  
 Bell, B. 258  
 Bereiter, C. 151  
 Bhatta, S. vi, 192, 265  
 bias 123, 209, 255, 311, 313  
 bias-filter 255  
 Bielaczyc, K. 278  
 Bigler, K. D. 69  
 Billington, R. vi  
 Binford, T. O. 55  
 Birnbaum, L. 25, 31, 32, 41, 60, 122, 155, 173, 270, 271, 294  
 BK (see background knowledge) 19, 26, 27, 28, 33, 59, 60, 62, 73, 77, 79, 107, 110, 112, 114, 116, 121, 122, 123, 127, 128, 147, 148, 162, 172, 178, 179, 190, 195, 198, 202, 209, 221, 247, 249, 250, 252, 272, 288  
 blame assignment (operationalized) 125  
 blame assignment (psychology) 230  
 blame assignment, see credit assignment 3, 8, 10, 11, 16, 26, 63, 68, 114, 122, 123, 124, 125, 131, 136, 137, 138, 140, 149, 154, 156, 158, 162, 164, 171, 178, 197, 198, 200, 214, 215, 216, 246, 252, 259, 264, 270, 271, 284, 288, 312, 380, 406  
 blame-assignment 149, 201, 216  
 blocked goals 60  
 Blocks World 130, 160, 161, 197  
 Blocks World operator 161  
 Bock, C. 267  
 Booker, L. B. 201  
 Boring, E. G. 8, 242  
 BORIS 220  
 Bounded-Cost EBG 248  
 boy who cried wolf 306  
 Bransford, J. D. 265  
 Brazdil, P. B. 264  
 Bredeweg, B. 107  
 Brewer, W. F. 255  
 BRIDGER 118  
 Brigham, M. C. 276  
 Brodley, C. 173, 197, 248  
 “brother-clobbers-brother” goal interaction (see Sussman’s anomaly) 157, 247  
 Brown, A. 108, 241, 265, 272, 280  
 Brown, L. 283  
 Buchanan, B. G. 173, 267, 268, 305, 313  
 Byrne, M. vi
- ## C
- caching 248  
 Callantine, T. J. 107  
 Carbonell, J. G. 10, 18, 84, 117, 118, 119, 134, 177, 211, 251, 268, 269, 270  
 Cardie, C. 220  
 Carey, S. 153  
 Carpenter, S. vi  
 cascade models 118  
 case acquisition 197, 248  
 case-based explanation 136  
 case-based introspection 20, 134, 135, 216  
 case-based learning 118

- case-based planning 386
- case-based problem solving 136
- case-based reasoning (see CBR) 3, 11, 12, 20, 27, 62, 69, 79, 114, 134, 136, 197, 257, 264, 265, 269, 271, 272, 278, 279, 386, 389, 403
- case-based understanding 20, 135
- cases 33
- CASTLE 271, 272
- category grouping 279
- causal invariants (for taxonomy of failure cause) 64
- CBR (see case-based reasoning) 134, 135, 151, 152, 179, 197
- CD 74, 177, 188, 201, 287
- CD representation 74, 101
- CD theory 68, 69, 71, 72, 75, 101
- Ceci, S. J. 272
- Celan, P. 3
- CELIA 136
- central processor 74, 75
- Chalasani, P. 118, 248, 269
- Chalmers, D. 293
- Chandrasekaran, B. 107
- Cheeseman, P. 196
- CHEF 156
- Cheng, J. 118, 248, 269, 271
- chess-playing program 303
- Chi, M. T. H. 220, 258, 278, 279, 301, 302, 305
- Chinn, C. A. 255
- chunking mechanism 117
- Clancey, W. J. 54, 267, 268, 305
- classification 256, 311
- classifier 312, 313
- CMU LISP tutor 230
- co-domain 75, 80
- cognitive model 230
- cognitive model of memory 177
- cognitive monitoring 273, 277
- cognitive predictions 251
- cognitive psychology 263
- cognitive science v, vi, 54, 105, 246, 298
- cognitive science community 306
- cognitive self-monitoring 277
- cognitive understanding 303
- cognitive vocabulary 68
- “cognitively homogeneous” terms 68
- Cognize 68, 90, 91, 93, 95, 96, 98, 251
- Cohen, S. M. 238
- Coleridge, S. T. 133
- Collins, G. C. 31, 41, 60, 122, 155, 173, 174, 270, 271
- combinatorial explosion of inferences 278
- compare and contrast strategy 256
- component theory 25
- componential frameworks 107
- composite IMXP 87
- composite meta-explanation 140
- comprehend 255
- comprehension 16, 34, 105, 109, 130, 218, 220, 273, 277, 283
- comprehension monitoring 90, 257, 279
- computational overhead 7, 243
- computational panacea 8
- CONC 74
- conceptual clustering 118
- conceptual dependency relationship 71
- conceptual dependency theory 68, 73
- conceptual processor 74
- conceptual stability 167
- conceptualization 68, 74, 75, 110
- conflict impasse 47
- conflict resolution 57
- connectionist nets 123
- consciousness 266, 293
- constituents of reasoning 51
- construction failure 87, 96
- constructive induction 117
- content 275, 285, 287
- content theory xxix, 25, 32, 33, 34, 37, 105, 287, 290
- content theory of introspective learning 33, 34, 35
- content theory of story understanding 32, 33, 35

context 266  
 contradiction 16, 29, 46, 48, 50, 56, 65, 72,  
     95, 98, 106, 121, 136, 140, 149,  
     156, 209, 256, 311  
 contributions 285  
 control 268, 269, 277  
 control knowledge 7, 267  
 control strategy learning 201  
 Converse, T. 60, 147, 160, 311  
 core IMXP 84, 90, 91  
 correct rejection 313  
 Cox, M. T. i, ix, 7, 8, 9, 11, 12, 16, 17, 18,  
     19, 20, 21, 22, 23, 27, 28, 41, 52, 74,  
     76, 79, 93, 114, 122, 123, 124, 147,  
     213, 238, 240, 255, 260, 264, 270,  
     278, 299, 300  
 Cox, B. M. iv  
 Cox, C. N. iv  
 Cox, J. S. W. vii  
 Cox, R. M. vi  
 credit assignment, see blame assignment  
     123, 312, 313, 405  
 Crockett, L. J. 304  
 cue elaboration 93, 279  
 Cullicover, P. 153  
 Cullingford, R. 26, 182, 220  
 cup domain 123  
 cup-domain 55  
 CYRUS 177

## D

Danyluk, A. P. 118  
 Davidson, H. 275, 279, 280  
 Davis, R. 18, 267, 268, 305  
 D-C-NODE 86, 115  
 D-C-Node 83, 84  
*De Trinitate* 263  
 deception 63, 201  
 Decide-Compute-Node 83, 85, 86  
 deciding what to learn xxix, 8, 10, 26, 122,  
     131, 149, 197, 209, 270, 271, 278,  
     284, 289  
 deciding what to learn (operationalized)  
     127  
 deciding what to learn (psychology) 230  
 decision model 81, 111  
 decision tree 200, 311  
 decision-analytic method 269  
 declarative knowledge 31, 301  
 declarative representation (see procedural  
     representation) xxix, 3, 7, 10, 30,  
     31, 34, 67, 68, 79, 154, 259, 265,  
     278, 285, 287, 301  
 deduction 117  
 deductive reasoning 266, 267  
 deductive theorem-proving 8  
 Deep Thoughts 293  
 default inheritance 268  
 degrees of freedom 312, 313  
 DeJong, G. 162  
 deKleer, J. 265, 270  
 Delclos, V. R. 274, 280  
 deliberate learning 241  
 deliberation cost 269  
 Dennett, D. C. 266, 293  
 dependency-directed backtracking 72  
 derivational analogy 10, 18, 84, 268, 378  
 Derry, S. J. 274, 302  
 Descartes, R. 67, 69, 304, 305, 306  
 descriptive theories 267  
 design 16  
 design stance 271  
 desire-psychology 301  
 desJardins, M. 311  
 Deuser, R. 275  
 Devaney, M. vi  
 device failure 259  
 diagnostic knowledge 238  
 discovery 255  
 discrimination net 136  
 Dixon, R. A. 275  
 DMAP 177, 207  
 domain ontology 25  
 domain physics 25  
 Domeshek, E. A. vi, 25, 51

Donnellan, M. W. 107  
 Dörner, D. 273, 274  
 Dostoyevsky, F. 42  
 Doyle, J. 18, 19, 72, 76, 77, 243, 266, 267, 269, 270, 305  
 Draper, D. 170, 296  
 “drawing a blank” 47  
 dream 254  
 Dunbar, K. 110  
 Dunlosky, J. vi, 50, 279, 280  
 Dyer, M. G. 220

## E

ease-of-learning judgement 276  
 EBG 162, 165, 167, 168, 197, 204, 206, 248  
 EBL 118  
 ecological self 305  
 Edelman, G. M. 293  
 Edwards, P. 90, 107, 139, 173, 206, 270  
 Egar, J. 108  
 eight-square puzzle 44  
 Eiselt, K. iii, vi  
 EITHER 256, 271  
 Elaboration 93  
 elaboration 130, 131, 255  
 electronics diagnosis 294  
 Elvis 111, 112, 114, 185, 186, 201, 202, 203, 211  
 Elvis World 181, 185, 196, 221, 222, 286, 289  
 emotion 69  
 emotions 68  
 environment 51, 62, 90, 123  
 episodes 7  
 episodic knowledge 55  
 epistemologically adequate 72  
 epistemology 30, 267  
 Epstein, W. 72, 242  
 erroneous association 53, 57, 58, 61, 64, 140, 164, 199, 206  
 erroneous knowledge 258, 260  
 error-based filter 209

errors of commission 42, 46, 47, 52, 55, 57, 62, 63, 64, 87, 95  
 errors of omission 42, 47, 52, 55, 57, 63, 64, 87, 95  
 Escher, M. C. 263  
 Etzioni, O. 118, 170, 248, 269, 293, 296  
 evaluation 21, 22, 214, 243  
 evaluation criterium 212, 220  
 Evens, M. 260  
 evolutionary theory of learning 41  
 execution cost 269  
 expect 254  
 expectation 44, 45, 49, 75, 90, 93, 95, 96, 98, 101, 107, 133, 277  
 expectation failure 27, 64, 95, 140, 143, 256  
 expectation-driven reasoning 65, 91  
 expected outcome 44, 45, 48, 49, 81, 95, 120, 312  
 expected utility 247, 313  
 experimentation 118  
 expert systems 108, 264, 267, 271, 305  
 expert-systems 305  
 EXPLAINS node 80, 81, 91, 137, 143  
 explanation 267, 303  
 explanation failure 29, 245  
 Explanation Game 303  
 explanation generation 35, 83, 109, 113  
 explanation pattern 73, 79, 80  
 explanation-based generalization 162, 197, 204, 279  
 explanation-based learning 117, 118  
 explanation-pattern theory 68, 75, 257

## F

Fabricius, W. V. 69  
 facet 191  
 factor analysis 69  
 failing negative 256  
 failing positive 256  
 failure 3, 4, 5, 7, 34, 35, 124, 128, 130, 136, 143, 151, 198, 269, 311

- failure characterization 137, 138, 140
  - failure definition 106
  - failure detection 120, 184
  - failure detection algorithm 121
  - failure-based understanding 41
  - failure-driven bias 312
  - failure-driven input bias 313
  - failure-driven learning 7, 41, 72, 106, 311, 313
  - false expectation 14, 16, 46, 47, 48, 50, 65, 96, 97, 98, 106, 121, 286, 311
  - false positive 312, 313
  - false-assignment category 256
  - falsifiability 212, 216
  - Fano, A. 258
  - Farrell, S. J. vi
  - feelings-of-knowing (see FOK) 50, 275, 276, 278
  - Fikes, R. E. 21, 32, 160
  - filler 191
  - filter condition 160, 162, 165
  - filters 209
  - Finin, T. 32
  - first-class objects 80, 81
  - Fischhoff, B. 8
  - Fisher, D. H. 220
  - Fixx, J. 156
  - FK (see foreground knowledge) 19, 26, 33, 59, 60, 73, 77, 79, 98, 107, 110, 128, 139, 178, 179, 190, 195, 209
  - Flavell, J. H. 272, 277
  - flawed behavior 53, 62, 201
  - flawed heuristic 53, 62, 251
  - FOK (see feelings-of-knowing) 275, 276, 278, 279
  - Foot, B. vi
  - Ford, K. M. 54
  - foreground knowledge 19, 26, 73, 76, 96, 107, 110, 112, 129, 130
  - foresight 254
  - foretell 254
  - forget 17, 90, 124, 278, 405, 408
  - “forget” predicate 73
  - forgetting 5, 18, 19, 29, 47, 59, 61, 68, 69, 72, 73, 74, 75, 76, 78, 79, 87, 88, 89, 95, 96, 107, 124, 206, 209, 247, 278, 283
  - Forgotten Goal 53
  - forgotten goal 61, 64
  - Forrest-Pressley, D. L. 220, 272, 277
  - Fox, S. 41, 69, 173, 270, 271
  - frame 177, 190, 191
  - frame form 190
  - frame variable 190
  - frame-type designator 191
  - Freed, M. vi, 22, 41, 122, 173, 174, 264, 270, 271
  - Freeman, D. 196
  - Freeman, P. vi
  - fully introspective multistrategy learning 215, 217, 219, 227
  - fully reflexive (non-introspective) multistrategy learning 217
  - function of learning 298
  - function of problem solving 298
  - function of understanding 296
  - function templates 267
  - functional theory 25, 155
  - FUNES 177
  - future research 216
- G
- Gardner, H. 54
  - Garner, R. 272, 277
  - Garza, A. G. 107
  - Gavelek, J. R. 220, 277
  - GEMINI 118
  - generalization 207, 247
  - generate-and-test 108, 110, 128
  - generic task 107
  - Genesereth, M. R. 267, 268, 305
  - Ghosh, S. 22, 127, 159, 178, 197, 385
  - Gil, Y. 251
  - Glasser, R. 220, 258
  - Glenberg, A. M. 72, 242



goal competition 111  
 goal conflict 247  
 goal frame definition 147  
 goal orientation 117, 151  
 goal representation 146  
 goal structure 146  
 goal taxonomy 146, 151, 249  
 goal value 269  
 goal-based scenarios 258  
 goal-based strategy 301  
 goal-driven learning 3, 152  
 goal-driven planning 9, 151  
 Goda, J. vi  
 Gödel, K. 263  
 Goel, A. K. 107, 108, 122, 173, 265, 270, 271  
 Goldberg, D. E. 201  
 Goldman, N. 18, 74, 101, 152  
 Goldstein, I. 267, 305  
 Gombert, J. E. 277  
 Govindaraj, T. 238  
 granularity of representation 72, 170, 279  
 Green, C. C. 153  
 Greeno, J. G. 295  
 Gruber, T. R. 32

## H

habitual behavior 296  
 Hale, C. R. vi, ix, x, 258, 260  
 Halford, G. S. 212  
 Hammond, K. J. 17, 41, 42, 44, 60, 128, 134, 147, 155, 156, 160, 270, 311  
 Handey, J. 153, 293  
 Hanks, S. 170, 296  
 Harley, S. 220  
 Harrington, C. 274, 280  
 Hayes, P. J. 30, 31, 54, 72, 73, 265  
 Hayes-Roth, B. 60  
 Hayes-Roth, F. 41, 60, 79  
 Heinlein, R. A. 293  
 Hendler, J. 22, 127, 159, 178, 197, 247, 385  
 Hertzog, C. K. vi, 275

heuristic classification 108  
 heuristic search 107  
 heuristics 7, 17, 30, 59, 62, 110, 151, 200, 209, 238, 244, 255, 256, 261, 278  
 hierarchical case-representations 245  
 hierarchical planner 159  
 hindsight 48, 49, 90, 98, 254  
 Hinrichs, T. R. vi, 134  
 hit 313  
 Hmelo, C. vi  
 Hofstadter, D. R. 263, 293  
 Holland, J. H. 201  
 hope 254  
 Horn-clause logic 257, 261  
 Horn-clause propositional logic representations 245  
 Horn-clause rules 56  
 Horty, J. 264  
 Hultsch, D. F. 275  
 human problem-solving 107  
 Hume, G. 260  
 Hunter, L. E. 9, 10, 12, 20, 31, 93, 114, 118, 122, 123, 147, 152, 155, 158, 167, 173  
 hypotheses 65, 212, 213, 289  
 hypothesis formation 120  
 hypothesis generation 110, 141  
 hypothesis space 110  
 hypothesis verification 35, 83, 109, 110, 113, 120  
 hypothesize alternative solution strategy 232

## I

ID3 248, 311  
 identify comprehension failure strategy 232, 234  
 ILT learning task 116  
 imagine 68, 254  
 Imitation Game 302, 303  
 IML algorithm 125, 126, 250  
 IML Theory 277

- IML theory 5, 11, 16, 19, 22, 23, 30, 32, 34, 65, 69, 75, 81, 107, 110, 117, 133, 136, 146, 147, 151, 178, 188, 207, 213, 229, 233, 236, 241, 242, 245, 246, 251, 256, 257, 258, 260, 261, 263, 267, 272, 275, 277, 278, 279, 280, 281, 285, 288, 289, 303, 311
- immediate memory 74
- impasse 14, 16, 18, 28, 46, 47, 48, 50, 59, 65, 87, 95, 96, 106, 121, 136, 204, 209, 251, 286, 311
- IMXP (see Introspective Meta-XP) 18, 20, 22, 27, 29, 34, 79, 84, 87, 90, 121, 125, 127, 135, 136, 137, 138, 139, 143, 149, 197, 198, 202, 204, 206, 213, 247, 249, 250, 288
  - structure specification 84
- IMXP ANOMALY-AND-BAFFLED 87
- IMXP-ANOMALY-AND-BAFFLED 198
- IMXP-ANOMALY-EXPLAINED 198, 202
- IMXP-BAFFLED-AND-RESOLVED 89, 198
- IMXP-EXPECTATION-FAILURE 198
- IMXP-NOVEL-SITUATION 198
- IMXP-NOVEL-SITUATION-ALTERNATIVE-REFUTED 143, 198
- IMXP-NOVEL-SITUATION-ALTERNATIVE-REFUTED-NO-ANOMALY 198
- IMXP-RETRIEVAL-FAILURE 198
- IMXP-SUCCESSFUL-PREDICTION 92, 198
- inappropriate goal 5
- incompleteness 55
- inconsistency 72, 267
- incorporation failure 87, 90, 95, 140, 143, 202
- incorrect context 53, 63
- incorrect domain knowledge 53, 55, 58, 140, 156, 199, 201
- incorrect index 78
- incorrect reasoning choice 201
- index implementation 192
- index learning 124, 164, 167, 168, 197, 248, 279
- index representation 192, 193
- indexing 51
- indexing problem 51, 106
- indexing vocabulary 32, 137
- induction 117
- inductive learning 165, 173, 238
- inductive policy 209, 313
- inference rules 267
- inferential expectation failure 84, 90
- Inferential Learning Theory 116
- information processing 276
- INGEST 201, 202
- input bias 209, 311, 313
- input failure 87, 90
- input generator 184
- insight 117, 254, 255, 275, 277, 302, 305
- inspect 254
- instrumental scene 182
- integrate by parts 54
- integration 294, 306
- intelligence 4, 275, 293, 306
- intelligence test 294
- intelligent learning environment 257, 259, 268
- intelligent learning environments 258
- intelligent tutoring systems 245
- intend 255
- intentional stance 266, 271
- interestingness 109, 110, 182, 188, 204, 295, 296
- interestingness heuristics 110
- INTERNAL-XP-NODES 80, 81
- interpretive case-based reasoning 135
- introspect 254
- introspection xxix, 7, 8, 10, 15, 19, 25, 33, 72, 81, 102, 111, 124, 133, 135, 138, 140, 214, 216, 218, 219, 242, 243, 252, 263, 264, 265, 266, 267, 268, 269, 272, 275, 279, 280, 281, 283, 284, 285, 287, 288, 289, 302,

303  
 introspection as objective inspection (observation) 242  
 introspective explanation 83, 151, 259  
 introspective knowledge 299  
 introspective learning 128, 135  
 Introspective Meta-Explanation Pattern (see IMXP) 101  
 Introspective Meta-XP 18, 79  
 introspective multistrategy learning xxix, 3, 10, 105, 115, 125, 213, 264, 290  
 introspective multistrategy learning algorithm 124  
 introspective question 139  
 introspective questioning 139, 206  
 INVESTIGATOR 118, 152  
 IQ 275  
 ISM 118, 248, 271  
 IULIAN 107, 270

## J

Jameson, A. 254  
 Johnson, H. M. 63, 201, 220, 278  
 Johnson, J. S. iv  
 Johnson-Laird, P. N. 265  
 JOL (see judgements-of-learning) 275  
 Jona, M. 258  
 Jones, E. K. 191, 299, 306  
 Jones, R. M. 41, 278  
 judgement of progress 277  
 judgements-of-learning (see JOL) 50, 275, 276  
 justification structure 266, 270, 271  
 justification trees 80

## K

Kambhampati, S. 22, 127, 159, 178, 197, 385  
 karma vi, 283  
 Kass, A. 20, 42, 75, 134, 137, 156, 270  
 Katz, B. 55

Kausler, D. H. 275, 276  
 Keane, M. T. 78  
 Kedar-Cabelli, S. 55, 162, 270  
 Keil, F. C. 153  
 Kell, A. 22  
 Keller, R. M. 55, 122, 162, 270  
 Kelly, J. 196  
 Kennedy, A. C. 69, 270  
 Kerner, Y. 75, 152  
 Kettler, B. 22, 127, 159, 178, 197, 385  
 keyword mnemonic strategy 276  
 keyword search 279  
 Klahr, D. 110  
 Kluwe, R. H. 274  
 Knoblock, C. A. 117, 118, 119, 269  
 knowledge acquisition assistant 108  
 knowledge acquisition goal 13, 20, 148  
 knowledge acquisition shell 108  
 knowledge acquisition strategy 5  
 knowledge dependencies 172  
 knowledge differentiation goal 21, 148, 149, 151, 158, 162  
 knowledge expansion goal 20, 148, 207  
 knowledge goal 83, 114, 296  
 knowledge how 31  
 knowledge organization 278  
 knowledge organization goal 21, 148  
 knowledge reconciliation goal 21, 148, 149, 151, 156, 158, 162, 165, 172  
 knowledge refinement goal 20, 148  
 knowledge sharing project 32  
 knowledge space 152  
 knowledge that 31  
 knowledge transmutations 121, 248  
 knowledge-based systems 107, 267  
 knowledge-level 23  
 knowledge-level theories 30  
 Kolodner, J. L. iii, v, vi, 41, 51, 60, 130, 134, 135, 152, 177, 192  
 Konolige, K. 264, 266  
 Koton, P. 134  
 Krinsky, R. 50  
 Krulwich, B. 41, 90, 122, 173, 270, 271

Kuokka, D. R. 79, 107, 118, 269

## L

Lachman, J. L. 278

Lachman, R. 278

Laird, J. E. 13, 117, 269

Langley, P. 201

Leake, D. B. vi, 10, 12, 20, 41, 42, 69, 75,  
122, 134, 137, 147, 173, 211, 259,  
270, 271, 273

learned rules 260

learning by observation 136

learning goal xxix, 3, 10, 11, 12, 13, 14, 20,  
84, 116, 117, 123, 124, 127, 137,  
146, 147, 148, 149, 151, 152, 158,  
162, 163, 209, 214, 216, 218, 243,  
260, 271, 272

learning goal metaphor 13

learning goal taxonomy 148

learning strategy 4, 7, 21, 143

learning strategy construction (psychology)  
230

learning-strategy construction 10, 15, 26,  
164, 171, 197, 248, 252, 270

learning-strategy construction (operation-  
alized) 127

learning-strategy construction problem xx-  
ix, 3, 7, 8, 12, 15, 108, 122, 157,  
158, 284, 285, 290

learning-strategy execution 197

least-commitment approach 159

Lebowitz, M. 165

Lee, M. S. 248

Lehnert 220

Lehnert, W. 220

Lenat, D. B. 79, 267, 305

Leonesio, R. J. 254

Lesh, N. 170, 296

Lewis, A. 220, 258, 273

LEX 54, 313

Lichtenstein, S. 8

Lieberman, D. A. 242

LISP 268

LISP learning 116, 151, 214, 230, 242, 253,  
286

LISP programming 4, 9, 116, 230, 233, 294

LISP troubleshooting domain 22

“list and induce” taxonomic method 65

literal 191

literal frame 190

Loebner Prize 303

Loebner, H. G. 302

logic 31, 73, 74, 75, 101, 264, 265, 266,  
267, 271, 280, 287

logical consistency 32

logical inconsistency 72

logical omniscience 72

logically complete 72

logically consistent 72

long-term memory 74, 75, 107

long-term store 19, 178

loose coupling 154, 172, 214

loud noises 110

Lovelace, E. A. 275, 276

Lowry, M. 55

Lyons, W. 263

## M

MacKinnon, G. E. 272

macro-level learning 218

Maes, P. 264, 268, 269

Magic Squares 124

Mahesh, K. vi

maintenance goal 146

making sense 303

Markovitch, S. 78, 177, 209

Marks, M. 60, 147, 311

Marsh, G. R. 275

Martin, C. E. 134, 177, 207

Martin, J. vi, 251

MAX 107

maximally general index 164

MBUILD 18, 74

MBR (see model-based reasoning)

- McCarthy, J. 30, 31, 72, 73, 220, 264, 265, 266, 267, 293  
 McClelland, J. L. 123  
 McDermott, D. 77, 267  
 McDermott, J. 107  
 McKay, D. 32  
 McLaren, B. M. 134  
 McNamara, T. P. 265  
 McRoy, S. W. 72  
 means-ends analysis 61, 124  
 MECH ix, x  
 Medin, D. L. 117, 251  
 Meehan, J. 22, 178, 184, 188  
 memory 275, 278  
 memory architecture 207  
 memory monitoring 276  
 memory organization package (MOP) 82  
 memory performance monitoring 275  
 memory reindexing strategy 5  
 memory search 270  
 memory search strategy 277  
 memory self-evaluation 275  
 memory strategy 275  
 mental model 265  
 mental simulation 67, 265  
 Meta-AQUA vi, xxix, 3, 10, 11, 12, 20, 21, 22, 23, 26, 27, 28, 29, 31, 32, 45, 56, 59, 62, 65, 77, 78, 87, 102, 110, 111, 112, 118, 133, 134, 136, 137, 138, 139, 140, 143, 149, 151, 152, 154, 155, 156, 158, 167, 168, 170, 171, 173, 177, 178, 179, 180, 181, 182, 184, 185, 186, 188, 190, 192, 194, 195, 196, 198, 200, 201, 202, 203, 204, 206, 207, 209, 212, 213, 214, 215, 218, 219, 221, 224, 225, 227, 229, 230, 231, 233, 234, 236, 241, 242, 243, 246, 248, 249, 250, 251, 253, 256, 257, 263, 269, 270, 271, 272, 278, 279, 281, 285, 288, 289, 294, 295, 303, 311  
 Meta-AQUA file system 181  
 Meta-AQUA flow of control 183, 200  
 metacognition 9, 22, 214, 218, 242, 267, 268, 272, 273, 275, 276, 277, 278, 279, 280, 281, 301, 302, 377, 379, 385, 386, 403, 408  
 metacognitive component 220  
 metacognitive knowledge 302  
 metacognitive monitoring 269, 279  
 metacognitive skill 274  
 metacomprehension 218, 220, 277  
 meta-declarative knowledge 301  
 meta-explanation 3, 137, 215  
 Meta-Explanation Pattern (see Meta-XP) 79  
 meta-explanation pattern 11, 79, 83, 288, 290  
 metaknowledge 7, 107, 264, 267, 278, 279, 299, 305  
 meta-level 276  
 metalevel 302  
 metalevel architecture 264  
 meta-level knowledge 267  
 meta-level process 108  
 meta-level processing 277  
 meta-level reasoning 268  
 meta-level tools 108  
 metalinguistic development 277  
 metamemory 5, 272, 275, 277, 280  
 metaphysical interpretation 73  
 meta-plane 108  
 meta-reasoning 10, 79, 81, 264, 268  
 metareasoning 267, 268, 284, 288  
 metarule 267, 268  
 meta-strategy 301  
 Meta-TS 23, 230, 238, 239, 241, 289, 294  
 meta-X definition 79  
 Meta-XP (see meta-explanation pattern) 11, 14, 18, 22, 27, 52, 75, 78, 79, 84, 94, 102, 125, 127, 133, 138, 140, 145, 155, 180, 213, 249, 253, 256, 270, 278, 285, 286, 288  
 Metcalfe, J. 290  
 Michael, J. 260  
 Michalski, R. S. 8, 12, 116, 117, 122, 165,

248  
 micro-index 192, 194  
 micro-level learning 217  
 Miller, D. L. 265  
 mind-body problem 299  
 Miner, A. C. 278, 279  
 MINERVA 178, 271  
 Minsky, M. L. 122, 123, 178, 264, 265, 299, 302  
 Minton, S. 17, 61, 117, 118, 119, 269, 270, 311  
 misinformation 5, 63  
 miss 312, 313  
 missing association 18, 53, 57, 58, 64, 78, 199, 200, 206, 256  
 missing behavior 53, 62, 64, 201  
 missing context 53, 63, 64  
 missing goal 53, 60, 64, 78  
     missing (retrieval) goal 78  
 missing heuristic 53, 62, 64, 251  
 missing index 18, 78  
 missing input 53, 63, 64  
 missing knowledge 258, 260  
 Mitchell, T. M. 54, 55, 118, 123, 153, 162, 201, 238, 269, 270, 313  
 MLOC 74  
 MOBAL 118  
 model 299  
 model-based reasoning 272  
 MOLGEN 108  
 monitoring 272, 274, 276, 277, 299  
 Mooney, R. 56, 162, 256, 270, 271  
 Moore, A. W. 248  
 Moore, R. C. 31, 266  
 Moorman, K. vi, 153  
 Morik, K. 118  
 motivational explanation 81  
 MTRANS 18, 74  
 MUC-3 (see Third Message Understanding Conference )  
 multi-category domain theory 256  
 multi-category theories 256  
 multi-level reasoning 108

multistrategy learning xxix, 8, 117, 119, 128, 271, 393, 395, 398  
 multistrategy reasoning 106, 107, 271  
 mumble function 188  
 Musen, M. 108  
 mutual-indexing schema 166

## N

Narayanan, S. vi, 17, 23, 238, 240, 260, 299  
 Nardi, D. 264  
 Narens, L. 241, 254, 269, 276, 277  
 natural language understanding 74  
 Neches, R. 32  
 negative classification 312  
 Neisser, U. 305  
 Nelson, T. O. 50, 241, 254, 269, 276, 277, 279, 280  
 Nersessian, N. J. iii, vi, 13  
 Newell, A. 13, 17, 25, 30, 47, 54, 107, 117, 124, 146, 195, 212, 268, 269, 294, 296, 299  
 Ng, E. 151  
 Nilsson, N. J. 21, 160  
 Nisbett, R. E. 242  
 NOAH 159  
 no-change impasse 47  
 noise 53, 63, 201, 312  
 Nolan, E. vii  
 non-incremental learning 196  
 non-introspective multistrategy learning 217  
 non-introspective reasoning 269  
 Nonlin 22, 127, 159, 167, 168, 178, 181, 197, 249  
 non-linear learning plan 289  
 non-linear planner 21, 159, 167, 249  
 non-linear planning 3, 134, 216, 284, 288  
 nonmonotonic knowledge 266  
 normative decisions 114  
 NORMATIVE-FILLER-VIOLATION 156  
 not remember 73, 74, 96  
 notice 255

novel situation 53, 55, 58, 64, 77, 78, 140,  
149, 164, 199, 206, 256

Noyes, C. R. 69

null hypothesis 216

## O

object-level 276

OCCAM 118, 185, 270

Oehlmann, R. vi, 90, 107, 139, 173, 206,  
270

off-line evaluation 275

omniscient 275

on-line evaluation 275

on-line memory self-evaluation 275, 276

ontological vocabulary 32, 69

ontology 25, 68, 69, 70, 90, 93, 98, 101,  
148, 299

operator preconditions 160

operator schemas 159

opportunistic learning 147, 171

opportunistic memory 190

opportunistic reasoning 28, 60, 83, 121,  
138

opportunistic strategy 4, 238

opschemas 159

oracle 200, 311

Osgood, R. 51

Osherson, D. N. 153

Ourston, D. 56, 256, 270, 271

outcome 44, 45

overgeneralization 202

overly general concept 312

overly general domain theory 56, 256

overly general indexes 57, 58

overly specialized concept 312, 313

overly specific domain theories 55, 56

overly specific indexes 57, 58, 59

Owens, C. 20, 41, 42, 49, 51, 59, 75, 128,  
134, 137, 154, 155, 156, 177, 192,  
270, 271

## P

PAGODA 311, 312

paired-associate 275

Palmer, S. E. 31

parallelism 172, 249

Park, Y. T. 178, 270, 271

partial credit 211, 289

Pascal programming 116, 268

passion taxonomy 67, 69

passive failure 209

Patel-Schneider, P. F. 32

Patil, R. S. 32

Pazzani, M. J. vi, 41, 118, 134, 185, 212,  
270

PDP nets 123

Pearce, M. vi

Penberthy, L. 122

percept 254

perception 254

perplexed 255

Persian Gulf oil embargo 312

person variables 273, 299, 305

perspective memory 61

Peterson, J. vi

PFXP 154, 155, 156, 157

philosophy 263, 266, 267, 273

physical explanation 83, 112

physical relation 75

Pirolli, P. 4, 22, 151, 218, 229, 230, 231,  
233, 253, 258, 278

Plan Failure eXplanation Pattern (see  
PFXP)

plan repair 156

plan to learn 272

planning failure 128, 154, 271

planning metaphor 168, 171, 247

Plaza, E. 173, 299

Pollock, J. L. 263, 273, 299

poor goal 53, 59, 60

poor selection 53, 61

Pope, A. vii, 263

portend 255

positive classification 312

possible worlds semantics 32  
 postdiction 276  
 pray 254  
 precept 254  
 predicate logic 32, 177  
 predicate representation 74  
 preference judgements 242  
 premonition 255  
 pre-performance estimates 276  
 pre-performance estimates of memory 276  
 prescience 255  
 prescriptive theories 267  
 presentiment 255  
 preservation goal 111  
 Pressley, M. 220, 276, 277  
 pretend 255  
 prevention goal 146  
 PRE-XP-NODES 80, 81, 137, 138, 139  
 primitives 69, 74, 87  
 principle of rationality 30  
 priority queue 180, 188  
 probability 269  
 problem elaboration 130  
 problem identification 275  
 problem of forgetting 52, 60  
 problem reformulation 35, 107  
 problem representation 107  
 problem understanding 295  
 procedural knowledge 31  
 procedural representation (see declarative representation) 67, 265, 301  
 process 285, 287, 294  
 process theory xxix, 25, 32, 33, 37, 105, 288, 290  
 process theory of introspective learning 33, 133  
 process theory of story understanding 32, 33  
 process theory of understanding 33  
 PRODIGY 18, 61, 118, 119, 269, 311, 378  
 production systems 117  
 project networks 249  
 propositional Horn-clause theories 256

PROTÉGÉ-II 108  
 protocol analysis 238  
 provably correct solutions 211  
 Provost, F. J. vi, 173, 313  
 psychological community 272, 305  
 psychological variables 273  
 psychology 264  
 Puerta, A. 108  
 Punch, W. F., III 107, 108  
 PUPS 274  
 puzzle-solving 274  
 Pylyshyn, Z. W. 117, 123, 153

## Q

question asking 277  
 question generation 279  
 question representation 139  
 question-answering strategy 277  
 question-driven story understanding 21, 83, 290  
 question-driven understanding 36, 111, 112, 113, 114, 129, 137, 138, 180  
 quicksort 268  
 Quilici, A. 173  
 Quinlan, J. R. 248, 311

## R

Ram, A. iii, v, vi, 7, 8, 9, 10, 11, 12, 16, 17, 18, 19, 20, 21, 23, 27, 41, 42, 45, 52, 60, 74, 75, 77, 80, 81, 82, 93, 110, 111, 114, 116, 122, 123, 124, 134, 137, 147, 152, 153, 167, 173, 178, 182, 191, 195, 211, 221, 238, 240, 248, 255, 260, 273, 278, 296, 299, 306  
 Ram Das 283  
 Raphael, T. E. 220, 277  
 RAPTER 270, 272  
 rationality 195  
 reactive 296  
 reactive behavior 108



- reading comprehension 72, 273, 277  
 reality monitoring 276  
 Reason, J. 41, 61  
 reasoning failure 45  
 reasoning strategies 267  
 reasoning trace 10, 83, 204, 270, 271  
 recall 69, 76, 90, 93, 94, 276, 278  
 Recker, M. iii, vi, 4, 22, 151, 229, 230, 231, 233, 253, 258, 278  
 recognition 69, 90, 93, 94, 107  
 reconsider 254  
 recovery 4, 128, 130, 155  
 Reder, L. M. 61, 278, 279  
 Redmond, M. A. vi, 134, 136, 156, 158, 173, 270  
 referential transparency 31, 32  
 referentially opaqueness 31  
 REFLECT 107  
 reflect 254  
 reflect on errors strategy 232  
 reflect over the problem solving strategy 4  
 reflection 33, 133, 135, 257, 263, 264, 268, 280  
 reflective blame-assignment algorithm 138  
 reflective knowledge 301  
 reflective strategy 232  
 rehearsal 279  
 rehearse 255  
 Reich, Y. 118  
 Reimann, P. 220, 258  
 Reinders, M. 107  
 reinforcement learning 123  
 Reiser, B. J. 230  
 reject impasse 47  
 religious-fanatic explanation 81  
 remember 69, 93, 254, 283  
 reminding 29, 69, 94  
 repair 3, 4, 7, 84, 108, 128, 130, 155, 156, 214, 264, 271  
 re-read solution strategy 232  
 re-read text strategy 4  
 reread text strategy 232  
 re-reading instructions strategy 121, 151  
 re-reading of text 279  
 resources limitations 247  
 retrieval failure 76, 87, 90, 96, 140  
 "retrieval list" 194  
 retrospect 254  
 review 254  
 reviewing an earlier example strategy 151  
 RFermi 270  
 Rieger, C. 18, 74, 101, 152  
 Riesbeck, C. K. 18, 74, 75, 101, 134, 152, 220  
 Riley, M. S. 295  
 Riloff, E. 220  
 Ringuette, M. 118, 248, 269  
 Rissland, E. L. 134  
 ROBBIE 271, 272  
 Roberts, J. vi  
 role 191  
 role-filler 191, 194  
 role-limiting methods 107  
 Rosenbloom, P. S. 13, 117, 269  
 Rosenfield, I. 293  
 Ross, B. H. 135  
 rote memorization 304  
 ROUTER 107  
 Rovick, A. 260  
 rule induction 118  
 rule models 267  
 rule-based systems 55  
 Rumelhart, D. E. 123  
 Rumiano, E. vi  
 Russell, C. vi  
 Russell, S. 269  
 Ryle, G. 31
- S
- Sacerdoti, E. D. 159  
 Salthouse, T. vi  
 SAM 182  
 Santamaria, J. C. vi  
 Sarrett, W. 212  
 Sartre, J. P. 245

- SBF model 201
- SBL 165
- Schaffer, C. 173, 248
- Schank, R. C. 18, 20, 26, 31, 41, 51, 68, 69, 71, 73, 74, 75, 101, 134, 137, 146, 152, 155, 177, 182, 220, 249, 258, 259, 295, 296, 303, 304, 305, 306
- schemata 267
- schematic knowledge 295
- Schlimmer, J. C. 118, 248, 269
- Schneider, W. 272, 277, 279
- Schoenfeld, A. H. 212
- Schooler, J. W. 8, 242
- Schrobe, H. 267, 305
- Schwanenflugel, P. J. 69
- science fiction 153
- scientific discovery 13, 15, 110
- Scott, P. D. 78, 177, 209
- script application 182
- script processing 33
- scripts 33
- Searle, J. 285, 289, 303, 305, 306
- second-order knowledge 67
- Seifert, C. M. 60, 63, 147, 201, 311
- selective experience 209
- selective superiority problem 197
- selective-experience filter 209
- selective-superiority problem 173
- self 273, 290, 302, 305, 306
- Self, J. 268, 276
- Self, M. 196
- self-comprehension 72
- self-description 305
- self-evaluation 279, 305
- self-explanation 218, 258, 278, 379, 398, 407
- self-explanation effect 220, 258
- self-knowledge 264, 270, 299, 304, 305
- self-model 265
- self-modification 305
- self-monitoring 272, 273, 277, 301
- self-preservation 293, 306
- self-selection of training examples 311
- self-understanding 7, 135
- semantic hierarchy 69
- Semantic information processing* 264
- semi-introspective multistrategy learning 216, 217, 219, 227
- sex 110
- Shankar, M. 122
- Shimamura, A. P. 290
- Shoham, Y. 264
- Siegler, R. S. 41, 61
- Siegmann, P. vi
- similarity judgement task 69
- similarity-based learning 117, 165
- Simina, M. 60
- Simon, H. A. 54, 107, 115, 116, 124, 146, 212, 294
- Simon, T. iii, vi, 212
- Simpson, R. L. 134, 152
- single-concept domain theory 256
- single-strategy learning 117, 119
- single-strategy system 21
- situated cognition 54
- situation assessment 130
- skeletal-plan refinement 108
- Skinner, B. F. 242
- Sleeman, D. 90, 107, 139, 173, 201, 206, 270
- slot 191, 194
- slot-filler pair 191
- Slovic, P. 8
- SMART 177
- Smith, A. D. vi
- Smith, B. C. 268
- Smith, R. G. 305, 378
- Smyth, B. 78
- Soar 13, 47, 96, 117, 218, 269, 390
- social psychology 273
- Soderland, S. 220
- solution checking 279
- solution evaluation 275
- Spear, N. E. 278
- speedup-learning 248
- sponsor-selector mechanism 108

- Stallman, R. M. 72  
 state-process dichotomy 69  
 statistical pattern matching 123  
 Steele, G. L. 270  
 Steels, L. 107  
 Stefik, M. 108, 268  
 Steier, D. M. 117  
 Stein, G. 67, 265  
 Sternberg, R. J. 275  
 Stevens, W. 177  
 Stich, S. P. 267  
 Stob, M. 153  
 story generator 184  
 story simulation 188  
 story understanding 22, 26  
 stranded motorist example 124  
 Strange Loops 263  
 strategy construction 271  
 strategy selection 17, 53, 61, 107, 114, 158  
 strategy selection models 118  
 strategy variables 273  
 STRIPS 21, 160, 197  
 strong methods 290  
 Stroulia, E. vi, 41, 122, 173, 201, 270, 271  
 structure-behavior-function model 201  
 student explanation failure 245, 259  
 student explanations 259  
 student model 268  
 Stutz, J. 196  
 success-driven learning 41, 250, 311  
 successful prediction 84, 91, 92, 93, 98  
 Suchman, L. 54  
 Sundheim, B. 220  
 SURF 218  
 surprise 14, 16, 46, 48, 49, 50, 65, 98, 99, 106, 121, 286, 311  
 suspect 254  
 suspend 255  
 suspended goal 51, 278  
 Sussman anomaly 160  
 Sussman, G. J. 12, 41, 72, 157, 247, 270, 288  
 Sutton, R. S. 123  
 SWALE 20, 137, 138, 156  
 Swanson, H. L. 274, 280  
 Sycara, E. P. 134  
 symbol-level theories 30
- ## T
- Tale-Spin vi, 22, 178, 179, 181, 184, 185, 186, 188, 190, 194, 196, 201, 203, 209, 219, 221, 222, 243, 252, 289  
 Tale-Spin file system 181  
 Tash, J. 269  
 Task Formalism 159, 160, 290  
 task variables 273  
 Tate, A. 22, 127, 159, 160, 247, 249, 290  
 taxonomy of failure causes 259  
 Taylor, W. 196  
 Tecuci, G. 8, 117  
 TEIRESIAS 268  
 teleological commitment 25  
 Tesler, L. G. 71, 101  
 text comprehension 101, 242  
 text understanding 277  
 textual information 277  
 Thagard, P. 13, 312  
*The moon is a harsh mistress* 293  
 Theo 118, 248, 269  
 theorem proving 8  
 theory of mind 267, 273, 301  
 theory revision 256  
 Third Message Understanding Conference (MUC-3) 220  
 Thompson, R. 274  
 Three Laws of Robotics 293  
 Thronesbery, C. 278  
 tic-tac-toe 124  
 tie impasse 47  
 tight coupling 154, 155, 156, 157, 172, 214, 264  
 time constraints 48  
 tip-of-the-tongue phenomena 276, 278  
 TIPS 107  
 Titchener, E. B. 242

TMXP (see Trace Meta-XP) 18, 20, 27, 34, 79, 84, 87, 91, 115, 120, 121, 125, 127, 135, 138, 139, 143, 147, 184, 197, 204, 213, 250, 271, 288  
 token 149, 162, 190, 191, 192, 195, 269  
 too many cooks spoil the broth 154, 157  
 toolbox models 118  
 Trace Meta-Explanation Pattern 101  
 Trace Meta-XP (see TMXP) 18, 79, 83, 84  
 transformation rule 270  
 troubleshooting 16, 23, 238, 241, 242, 258, 259, 260  
 truth value 76, 143  
 Tu, S. 108  
 Tulving, E. 31  
 Turing machine 265  
 Turing Test 302, 303, 304  
   (genesis by Descartes) 304  
 Turing, A. 47, 265, 294, 302, 303, 304, 305, 306  
 Turner, R. 134  
 tutoring 118  
 TWEAKER 156  
 type 149, 162, 190, 191, 269

## U

Ullman, J. D. 30  
 uncertainty-based filters 209  
 unexpected success 14, 16, 46, 49, 50, 65, 98, 100, 106, 121, 286, 311  
 UNIMEM 165  
 unsupervised condition 160, 165, 167  
 URL vii, 32, 185, 197  
 use-when condition 162, 165  
*USS Vincennes* 312, 313  
 Utgoff, P. E. 54, 201, 313

## V

value facet 191  
 van de Velde, W. 299  
 van Someren, M. 299

VanLehn, K. 41, 278, 299  
 variable bindings 83  
 Veloso, M. 18, 84, 118, 119, 134, 177, 270  
 violence 110  
 vocabulary 25, 30, 32, 35, 68, 69, 72, 84, 90, 98, 101, 138, 247, 254  
 volitional explanation 83, 112  
 volitional role relation 75  
 volitional XP 82, 195

## W

Waller, T. G. 272  
 Walnut Cove vii, 5, 7, 42, 47  
 Warfield, T. A. 267  
 Waterman, D. A. 79  
 Watson, J. B. 242  
 weak methods 290  
 Wefald, E. 269  
 Weinert, F. E. 241, 279  
 Weinstein, S. 153  
 Weintraub, M. A. 122  
 Weld, D. 170, 293, 296  
 well-formed formulae 31  
 Wellman, H. M. 272, 273, 277, 278, 279, 299, 301, 302, 305  
 Wexler, K. 153  
 Wilensky, R. 45, 71, 81, 111, 178, 191, 220, 247, 294  
 Wilkins, D. C. 178, 270, 271  
 Wilkinson, A. C. 72, 242  
 Williamson, M. 170, 296  
 Wilson, T. D. 8, 242  
 Winograd, T. 67  
 Winston, P. H. 55  
 wish 254  
 wishful thinking 254  
 Wisniewski, E. J. 117, 251  
 Wolff, J. S. vii  
 wonder 254  
 working memory 19, 74, 76, 77, 96, 178  
 world knowledge 299  
 World Wide Web vii, 32, 185, 197

write down solution strategy 232

## X

XP 33, 75, 79, 80, 81, 84, 91, 137, 143, 152,  
164, 190, 195, 206, 207

XP application 20, 62, 114, 137, 138

XP failure types 156

XP theory 75

XP-ASSERTED-NODES 80, 81, 137, 138,  
139

XP-GOAL-OF-OUTCOME->ACTOR 76,  
190, 195

## Y

Yang, C. 220

you can catch more flies with honey than  
you can with vinegar 157

Yussen, S. R. 272

## Z

Zeichick, A. L. 302

Ziolko, A. vi



